Natural Language Processing for Business Documents

Lefteris Loukas

Ph.D. Thesis

Department of Informatics
Athens University of Economics and Business

Supervisors:

Ion Androutsopoulos, George Paliouras
and George Leledakis

2025

Abstract

Natural Language Processing (NLP) for business and finance-related documents (Hahn et al., 2018; Chen et al., 2022) is an expanding research area applying computational techniques to text such as company filings, analyst reports, and economic news. These documents present unique challenges due to specialized vocabulary (El-Haj et al., 2019), the critical role of numerical data, distinct syntactic structures, and domain-specific semantics. These issues are compounded by broader difficulties, including processing large volumes of unstructured public data and deploying language models cost-effectively, especially for resource-limited organizations like small-medium enterprises (SMEs). Addressing these challenges is crucial for applications ranging from fraud detection (Goel and Gangolly, 2012), long-form summarization (Cao et al., 2024), and information extraction to financial question answering (Maia et al., 2018).

This thesis aims to advance applied NLP and the use of business documents for real-world tasks by addressing current industry challenges across different layers of artificial intelligence: more specifically, across the data, application, and deployment layer, with a consistent focus on resource-constrained environments. We tackle three main research questions: (1) How can open-access, unstructured business documents be effectively leveraged for NLP? (2) How can current NLP methods that utilize deep learning (DL) techniques be adapted and extended to create business value in tasks like automatic document tagging, considering the nuances of financial language, particularly its heavy reliance on numerics? (3) For common industrial text classification tasks, what are the most accurate and cost-efficient approaches in resource-limited settings? We investigate the latter question by focusing on a real-world use case of intent recognition from customer dialogues, comparing BERT-based models and Large Language Models (LLMs) and optimizing LLM deployment for cost.

Addressing these questions, we first focus on "democratizing" the access to business documents by developing EDGAR-CORPUS, the largest publicly available financial NLP corpus in English, domain-specific word embeddings (EDGAR-W2V) which outperform alternatives,

and EDGAR-CRAWLER, an open-source software toolkit for financial data extraction with hundreds of users from academic researchers to FinTech practitioners and web developers. Then, we introduce XBRL tagging, a real-world NLP task, where we compile FiNER-139, the first dataset for this task, and find that LSTM models can outperform BERT due to issues of the transformer architecture (and its standard tokenization technique) with numeric token fragmentation. After benchmarking different methods, we then propose a novel tokenization technique using pseudo-tokens for TRANSFORMER models that significantly improves performance on numeric-first tasks, leading to the release of new state-of-the-art BERT models (SEC-BERT), specially created for the finance domain. Finally, for cost-efficient intent recognition, we conduct a comprehensive benchmarking study on the Banking77 dataset (Casanueva et al., 2020), showing that smaller BERT-based models can be more effective and economical than LLMs, requiring only slightly more annotated training data than the LLMs. We also showcase "Dynamic Few-Shot Prompting", a Retrieval-Augmented Generation (RAG) based method that drastically reduces LLM inference costs while maintaining high accuracy, and explore the utility of synthetic data generation.

This thesis makes several key resources publicly available: the EDGAR-CORPUS, EDGAR-W2V embeddings, the EDGAR-CRAWLER software, the FiNER-139 dataset, the family of the SEC-BERT models, as well as a curated subset of the Banking77 dataset, containing expert-selected examples for each intent class, which we show to be crucial for achieving high performance in few-shot learning scenarios.

The work of the thesis constitutes a significant contribution towards advancing industrial NLP on business documents by providing foundational open-source resources, novel methodologies for handling the unique characteristics of financial text, particularly numerical data, and practical, cost-effective strategies for deploying advanced NLP solutions in real-world financial applications, especially benefiting small-to-medium enterprises.

Acknowledgments

Finishing a PhD degree is a long and complex process, both in terms of time and transformation. I feel it takes time for one to realize truly what an experience this has been. I am sure I will discover more about its impact in the weeks or months to come, maybe even years. But since I need to submit this document now, I must, in this current moment, express my gratitude to the people who made this journey possible. Therefore, followingly, I (try to) thank the people who helped me make it through this experience.

First of all, I would like to thank my supervision committee for supporting me in this journey. Most importantly, I would like to thank Ion Androutsopoulos for supporting me and guiding me through this opportunity. Thank you for pushing me to always aim higher while being a humble and real researcher. I am deeply grateful for your trust, for you allowing me to continue this work after a long break, and for your patience during the whole period. My collaboration with Ion has shaped not only my understanding of AI but, more importantly, my way of thinking. Thank you for that.

Secondly, I would like to thank George Paliouras (as well as Eirini Spyropoulou) from the National Center for Scientific Research "Demokritos" and the AI Document Intelligence Center of Excellence there, who were among the first ones to believe in me (professionally) and gave me the opportunity to work in this field. Their valuable feedback and support helped me greatly in finding the right paths. This work would not have been possible at all without them. I would also like to thank George Leledakis who accepted to join this committee and offered valuable insights with respect to Natural Language Processing and his expertise in financial applications.

I also owe special thanks to some other colleagues at NCSR "Demokritos", where this journey began. I owe a huge thanks to Makis Malakasiotis, who acted as a fourth unofficial supervisor in this thesis, being present in almost every stage of this journey through advice or hands-on work when needed. You know that this thesis would not be right here right now without you. Also, I'd like to thank Manos Fergadiotis for the hands-on help I had during my

first years of my PhD, teaching me some of the best practices out there, which stayed with me ever since. Thank you, friends, both for your help, your time, as well as your tolerance. I would also like to thank Nikolaos Manginas and Kostantinos Bougiatiotis for all the fun times we had together at NCSR "Demokritos" (and especially all the football/basketball we played), making this journey better. We will do it again. Possibly in the same fields.

Additionally, I would like to thank my colleagues at helvia.ai, namely Stavros Vassos, with whom I reconnected after many years. After spending some time working in AI (outside of NLP), I had the opportunity to collaborate with Stavros on projects at the intersection of Applied AI and Natural Language Processing, particularly as Large Language Models were emerging. Some of the things we worked on ended up shaping this thesis and were instrumental in allowing me to resume and complete my degree. Thank you for your trust and for showing me how much I enjoy pragmatic, applied R&D in computer science. Also, a huge thanks to my buddies that helped me hands-on at helvia.ai: Ilias Stogiannidis and Odysseas Diamantopoulos Pantaleon. I wish you both the best in your own PhD journeys in Edinburgh, and above all, I hope you have a great time together along the way.

I would also like to thank the following colleagues in no particular order: Fabian Billert, Spyros Barbakos, Elias Zavitsanos, Dimitris Mavroeidis, George Giannakopoulos.

For sure, it is no secret that balancing your PhD ambition with a (normal) life is hard, totally hard. I owe a big thank you to Klajdi Bodurri, Konstantinos Kanellis, Panos Nikitakis, Effie Vaggeli, Giannis Goulias, all my buddies from Larissa, Valina, Mike, Gio Kay and Simon S., for helping me keep my balance and continue work towards my dreams, each one in your unique way, even from afar sometimes. And especially, to Holly: thank you for teaching me what true support means. Thank you for listening to me and for helping me. This thesis would not have been possible without your sincere love and understanding.

Last, but not least, I would like to thank my parents Giannis and Christina, as well as my (big) sister Konstantina, for always doing as much as they can to help me in every possible way. I am forever indebted to them for their endless patience. You gave me everything you could for my early beginning and allowed me to move forward and set new and bigger goals. Thank you.

Contents

Abstract iii

Acknowledgments v

Table of Contents vii

List of Thesis Publications xi

List of Thesis Resources xv

1 Introduction 1

1.1 Overview of this thesis 1

1.2 Why is language in the financial domain any different? 3

1.3 Contributions 4

1.4 Outline of the remainder of this thesis 7

2 Open-Source Software and Resources 9

2.1 Introduction 9

2.2 Contributions 13

2.3 Related Work 14

2.4 EDGAR-CRAWLER 17

2.4.1 Software Architecture 17

2.4.2 Configuration and Further Examples 21

2.4.3 JSON Output 22

2.4.4 Impact in other NLP and AI work 23

2.4.5 Parsing Accuracy Tests 24

2.4.6 Human-in-the-Loop Maintenance with AI coding assistants 26

2.5 EDGAR-CORPUS 28

2.5.1 Creation process of EDGAR-CORPUS 30
2.5.2 Word Embeddings (EDGAR-W2V) 30
2.5.3 Experiments 32
2.6 Conclusions 35

3 Numeric Entity Recognition for XBRL Tagging 39
3.1 Introduction 39
3.2 Contributions 41
3.3 Related Work 42
3.3.1 Entity Extraction 42
3.3.2 Named Entity Recognition in Finance 43
3.3.3 Numerical Reasoning 44
3.4 Task and Dataset 44
3.5 Baseline Models 46
3.6 Baseline Results 48
3.7 Numeric Fragmentation in BERT 51
3.8 In-domain Pre-training 52
3.9 Improved BERT Results 53
3.10 Experimental Setup 54
3.11 Additional Experiments 55
3.11.1 Subword pooling 55
3.11.2 Subword BiLSTM with [NUM] and [SHAPE] 56
3.11.3 A Business Use Case 57
3.12 Error Analysis 58
3.13 Impact in other scientific studies 60
3.14 Can modern LLMs solve this task through prompting? 62
3.15 Do the masking strategies affect LLMs? 63
3.16 Conclusion 65

4 Resource-Limited Intent Recognition in Banking 69
4.1 Introduction 69
4.2 Contributions 70
4.3 Related Work 72
4.4 Task and Dataset 74

4.5 Methodology . . . 75
4.5.1 Fine-tuning BERT-based models . . . 75
4.5.2 Few-Shot Contrastive Learning with SetFit . . . 76
4.5.3 In-Context Learning . . . 77
4.5.4 Human Expert Annotation for Robustness . . . 78
4.6 Experiments & Results . . . 78
4.6.1 Experimental Setup . . . 78
4.6.2 Hyperparameter tuning in BERT-based models . . . 79
4.6.3 Results . . . 80
4.7 Cost-Effective LLM Inference using RAG . . . 83
4.7.1 Cost Analysis . . . 83
4.7.2 Dynamic Few-Shot Prompting with Retrieval-Augmented Generation (RAG) . . . 83
4.8 LLMs for Data Generation in Low-Resource Settings . . . 87
4.9 Error Analysis . . . 89
4.9.1 Errors in LLMs . . . 89
4.9.2 Errors in BERT-based models . . . 90
4.9.3 Addressing Overlapping Labels in Future Work . . . 90
4.10 Dynamic Few-Shot Prompting in other studies . . . 91
4.11 Conclusion . . . 92

5 Conclusions, Limitations and Future Work 97
5.1 Summary of work and conclusions . . . 97
5.2 Limitations . . . 99
5.3 Future Work . . . 99

Bibliography 103

Appendix A 125

Appendix B 133

List of Figures xvi

List of Tables xxi

List of Thesis Publications

Relevant to Chapter 2: Open-Source Software and Resources

- L. Loukas, M. Fergadiotis, I. Androutsopoulos, and P. Malakasiotis. EDGAR-CORPUS: Billions of Tokens Make The World Go Round. In *Proceedings of the Third Workshop on Economics and Natural Language Processing*, pages 13–18, Punta Cana, Dominican Republic, Nov. 2021c. Association for Computational Linguistics. doi: 10.18653/v1/2021.econlp-1.2. URL https://aclanthology.org/2021.econlp-1.2

- L. Loukas, F. Billert, M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos. EDGAR-CRAWLER: From Raw Web Documents to Structured Financial NLP Datasets. In *Companion Proceedings of the 32nd ACM Web Conference 2025 (WWW 2025)*, 2025a. https://github.com/lefterisloukas/edgar-crawler

Relevant to Chapter 3: Numeric Entity Recognition for XBRL Tagging

- L. Loukas, M. Fergadiotis, I. Chalkidis, E. Spyropoulou, P. Malakasiotis, I. Androutsopoulos, and G. Paliouras. FiNER: Financial Numeric Entity Recognition for XBRL Tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4419–4431, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.303. URL https://aclanthology.org/2022.acl-long.303

- L. Loukas, E. Spyropoulou, P. Malakasiotis, M. Fergadiotis, I. Chalkidis, I. Androutsopoulos, and G. Paliouras. System and method for automatically tagging documents. In *United States Patent and Trademark Office (USPTO) Patent US17/873,932*, 2021e

- L. Loukas, E. Spyropoulou, P. Malakasiotis, M. Fergadiotis, I. Chalkidis, I. Androutsopoulos, and G. Paliouras. System and method for automatically tagging documents. In *European Office (EPO) Patent Application 21 386 048.9*, 2021d

- L. Loukas, E. Spyropoulou, P. Malakasiotis, M. Fergadiotis, I. Chalkidis, I. Androutsopoulos, and G. Paliouras. System and method for automatically tagging documents. In *World Intellectual Property Organization (WIPO) Patent Application WO2023006773A1*, 2021f

Relevant to Chapter 4: Resource-Limited Intent Recognition in Banking

- L. Loukas, I. Stogiannidis, P. Malakasiotis, and S. Vassos. Breaking the Bank with ChatGPT: Few-Shot Text Classification for Finance. In *Proceedings of the Fifth Workshop on Financial Technology and Natural Language Processing and the Second Multimodal AI For Financial Forecasting*, pages 74–80, Macao, 20 Aug. 2023b. URL https://aclanthology.org/2023.finnlp-1.7

- L. Loukas, I. Stogiannidis, O. Diamantopoulos, P. Malakasiotis, and S. Vassos. Making LLMs Worth Every Penny: Resource-Limited Text Classification in Banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 392–400, New York, NY, USA, 2023a. Association for Computing Machinery. ISBN 9798400702402. doi: 10.1145/3604237.3626891. URL https://doi.org/10.1145/3604237.3626891

Other relevant work

- L. Loukas, N. Smyrnioudis, C. Dikonomaki, S. Barbakos, A. Toumazatos, J. Koutsikakis, M. Kyriakakis, M. Georgiou, S. Vassos, J. Pavlopoulos, and I. Androutsopoulos. GR-NLP-TOOLKIT: An Open-Source NLP Toolkit for Modern Greek. In *Proceedings of the 31st International Conference on Computational Linguistics: System Demonstrations*, pages 174–182, Abu Dhabi, UAE, Jan. 2025b. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-demos.17/

  An NLP toolkit for Greek. I started contributing to this open-source repository while working at helvia.ai*, since the company required processing Greek business documents and no suitable toolkits were available.*

- L. Loukas, K. Bougiatiotis, M. Fergadiotis, D. Mavroeidis, and E. Zavitsanos. DICE at FinSim-3: Financial Hypernym Detection using Augmented Terms and Distance-based Features. In *Proceedings of the Third Workshop on Financial Technology and*

Natural Language Processing, pages 40–45, Online, 19 Aug. 2021a. -. URL https://aclanthology.org/2021.finnlp-1.7

A summer hackathon where we competed in this shared task and finished 4th out of 20. The task was about detection of terms in a financial ontology, which laid some groundwork about the XBRL Tagging work in Chapter 3.

- S. Vassos, S. Goudelis, D. Balaouras, G. Vitalis, V. Nakos, G. Pigka, L. Tsagkli, Z. Tsionas, A. Chasanis, S. van de Burgt, M. Pors, S. Papadoudis, and L. Loukas. Now I know! Empowering Voters with RAG-enabled LLMs to Eliminate Political Uncertainty. In *Proceedings of the 13th Hellenic Conference on Artificial Intelligence*, SETN '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400709821. doi: 10.1145/3688671.3688784. URL https://doi.org/10.1145/3688671.3688784

  A summer project I undertook while at helvia.ai. We deployed a chatbot at www.toraksero.gr. This tool employed RAG to assist in comparing political viewpoints and provided grounded citations, a feature not available from LLM vendors like OpenAI/-Google or frameworks such as langchain/llama-index at that time. This project laid the groundwork for the future RAG-related contributions detailed in Chapter 4.

List of Thesis Resources

Annotated Datasets

- FiNER-139 dataset: a dataset of 1.1 million sentences from U.S. public company financial reports, annotated with 139 fine-grained entity types based on XBRL (eXtensive Business Reporting Language). Unlike traditional NER tasks with a few broad categories, FiNER-139 focuses on context-dependent numeric tokens and introduces automated XBRL tagging as a novel entity extraction challenge for the financial domain.
  Link: `https://huggingface.co/datasets/nlpaueb/finer-139`

- Banking77 representative samples: Contains a total of 231 expert-selected representative samples, which we found out to have super performance in few-shot classification compared to randomly selected ones.
  Link: `https://huggingface.co/datasets/helvia/banking77-representative-samples`

Corpus

- EDGAR-CORPUS: The biggest (English) collection of financial/business NLP documents, with more than 6 billion tokens inside. Link: `https://huggingface.co/datasets/eloukas/edgar-corpus`

Word Embeddings

- EDGAR-W2V: 200-dimensional English financial word embeddings trained with the skip-gram model (WORD2VEC) on hundreds of thousands of business/financial documents from SEC'S EDGAR.
  Link: `https://zenodo.org/records/5524358`

Software

- EDGAR-CRAWLER: Open-source software which downloads SEC's EDGAR filings and parses them to clean JSON data. An early verison of it was used to generate EDGAR-CORPUS. Still maintained, and also extended to support more types of filings. Currently at 380+ Github stars. Link: `https://github.com/lefterisloukas/edgar-crawler/`

Pre-trained Language Models

- SEC-BERT: A family of English BERT models for the financial domain.
  Link for SEC-BERT-BASE: `https://huggingface.co/nlpaueb/sec-bert-base`
  Link for SEC-BERT-NUM: `https://huggingface.co/nlpaueb/sec-bert-num`
  Link for SEC-BERT-SHAPE: `https://huggingface.co/nlpaueb/sec-bert-shape`

Chapter 1

Introduction

1.1 Overview of this thesis

Natural Language Processing (NLP) for finance is a growing research area, as evidenced by numerous workshops and conferences in the last years (Hahn et al., 2018; Chen et al., 2022; Mehta and Wang, 2024), as well as the recent Special Interest Group on Economic and Financial Natural Language Processing from the Association of Computational Linguistics (ACL)[1].

A large number of firms, including start-up companies and medium-sized enterprises, already operate in the NLP market, which is projected to be valued at over 50 billion dollars by 2026[2] and over 300 billion dollars in 2035[3]. Such organizations target a wide variety of use cases that are currently challenging, with some of them being, for example, fraud detection (Goel and Uzuner, 2016; Zavitsanos et al., 2022), information extraction from financial documents (Salinas Alvarado et al., 2015; Francis et al., 2019; Hampton et al., 2015, 2016), virtual assistants for financial question answering (Maia et al., 2018), and others.

There are many reasons why financial NLP applications are currently relatively immature and underserved and they are still being researched. One reason is that, although data from public regulatory sources may be available, the large volume of unstructured information often makes it difficult to process. On the other side, specialized information about businesses is difficult to get access to, due to its proprietary nature. Lastly, while in recent years we have seen the release of powerful foundational Large Language Models (LLMs) by providers like OpenAI, Google, Anthropic with great results on most benchmarks, these benchmarks have

[1] https://sigfintech.github.io/

[2] https://finance.yahoo.com/news/global-natural-language-processing-market-121900452.html, https://www.ecb.europa.eu/press/fie/box/html/ecb.fiebox202406_08.en.html

[3] https://www.rootsanalysis.com/natural-language-processing-market

not been focused on financial use cases until recently (Xie et al., 2024). Furthermore, these benchmarks have typically been developed with the support of unlimited credit grants and do not consider the cost-efficiency of the models, making it difficult to develop and deploy financial NLP systems that are both effective and economically viable in real-world industry settings.

The objective of this dissertation is to advance Natural Language Processing within the financial domain by addressing practical industry challenges through applied research. The work is structured around three core layers of the AI stack: the data layer, the application layer, with some focus also on the deployment layer. A key focus of this work is ensuring that the proposed methods can be practically applied in resource-constrained environments, enabling small-to-medium enterprises, such as startups, to adopt and benefit from the findings in a cost-efficient manner.[4]

More specifically, in this thesis we aim to answer the following three research questions:

- **RQ1:** How can we **use existing open-access business documents** in financial NLP to overcome the **data availability problems** in this domain? How can we collect and **transform unstructured business documents into a structured** format to be integrated in NLP pipelines? Once a foundational resource (NLP corpus) is created, what is their tangible value and utility for downstream financial NLP tasks? Can a **large-scale NLP corpus** derived from these documents be used to train **domain-specific word embeddings**? Apart from a corpus and the embeddings, which are static resources, can we make researchers' availability to recent financial NLP data as easy as possible, through an **open-source software**?

- **RQ2:** How can we create actual business value by adopting NLP and DL methods? What **real-world task** can we **solve** by **adopting and extending the previous foundational resources of RQ1**? **How well do current NLP techniques perform in automatic document tagging?** Does the **financial language** or writing **affect** the **performance** of current methods? How can we **extend** these **NLP methods to be more effective**?

- **RQ3:** What are the most **accurate and cost-efficient deployment strategies** for common **industrial text classification** tasks in **resource-limited settings**? Should one use

[4]Part of this research has been conducted during work at `helvia.ai`, a small-medium enterprise operating in the conversational AI/NLP domain.

BERT-based models or more recent and much larger **LLMs** (Large Language Models)? Given that many state-of-the-art LLMs are accessible via **commercial APIs** where prompting is the primary interaction method, our investigation focuses on in-context learning (ICL). **Which LLMs to use** and how many samples should we use during prompting? How can we make **in-context learning (ICL) less expensive** for a firm using cloud-based LLM providers? We explore this question through the case study of intent recognition from **real-world customer dialogues**[5].

1.2 Why is language in the financial domain any different?

Similar to other specialized fields like legal or biomedical studies, financial text (e.g., company filings, analyst reports, economic news, regulatory documents) exhibits distinct characteristics when compared to general-purpose text. These include a specialized vocabulary, particular syntactic constructions, and semantics deeply rooted in extensive domain-specific knowledge (El-Haj et al., 2019). Within finance, there are various genres of writing. For instance, there is the formal language of regulatory filings and annual reports, the analytical language of investment research, and the more lively and engaging language of financial news and market commentary. This work focuses on the English language and data from the financial regulatory filings of the United States, as well as real-world customer support dialogues. Followingly, we also describe how financial English differs from standard English, particularly in its vocabulary:

Financial terminology. Analogous to other specialized domains, financial English incorporates technical terminology that is not commonly understood by the general public. Examples include terms like ‘derivatives’, ‘amortization’, ‘EBITDA’ (Earnings Before Interest, Taxes, Depreciation, and Amortization), ‘bull market’, ‘quantitative easing’, ‘hedging’, and ‘arbitrage’.

In-domain use of words. Many ordinary words acquire specific, distinct meanings within the financial sector. For example, ‘exposure’ typically refers to the amount of money an investor or institution stands to lose. The term ‘leverage’ often denotes the use of borrowed capital to increase the potential return of an investment. ‘Float’ can refer to the number of a

[5]This specific use case was motivated by a direct industrial challenge encountered during my work at `helvia.ai`, a conversational AI company, where finding a cost-efficient yet accurate solution for a banking client was a primary objective.

company's shares available for trading by the public. A 'security' is a tradable financial asset, and 'premium' can mean the price of an option contract or the amount paid for an insurance policy.

Use of acronyms, initialisms, and jargon. While direct borrowing from classical languages might be less prevalent than in law, financial English is saturated with acronyms, initialisms, and jargon that act as shorthand for complex concepts but can be opaque to outsiders. Examples include 'IPO' (Initial Public Offering), 'SEC' (Securities and Exchange Commission), 'LIBOR' (London Interbank Offered Rate), 'SWIFT' (Society for Worldwide Interbank Financial Telecommunication), 'P/E ratio' (Price-to-Earnings ratio), and terms like 'short selling' or 'dark pool'. These are integral to communication in the financial sector.

Heavy use of numerics. Financial language is densely populated with numerical data. Numbers are fundamental for quantifying performance, risk, valuation, and trends. These appear as absolute values (e.g., revenues, profits, asset values), percentages (e.g., growth rates, interest rates, market share), ratios (e.g., P/E ratio, debt-to-equity ratio), monetary amounts (often with specific currency symbols like $, €, £), and dates (e.g., fiscal year ends, reporting periods, maturity dates). The interpretation of these numbers often requires significant contextual understanding beyond the simple recognition of the numeric tokens themselves.

1.3 Contributions

The contributions of this thesis can be summarized as follows. We start with the contributions related to the open-source data and software (Chapter 2).

The largest publicly-available financial NLP Corpus (Chapter 2): We compile and publish a new corpus (EDGAR-CORPUS), the biggest corpus available in financial NLP (in English). This can be used for pre-training language models or word embeddings.

Domain-specific word embeddings (Chapter 2): We introduce new domain-specific English word embeddings trained on EDGAR-CORPUS that better capture the financial language, compared to generic word embeddings (Pennington et al., 2014b; Bojanowski et al., 2016) trained on data derived from the web, as well as other static financial embeddings. For short-text tasks like ontology classification, the word embeddings of this thesis also surpass modern contextual LLM-derived embeddings. Furthermore, their computational efficiency makes them a strong baseline, enabling easy deployment on standard CPUs and making them

ideal for resource-constrained environments where acquiring expensive GPUs is difficult.

New software to download and parse financial data in a standardized form (Chapter 2): We develop EDGAR-CRAWLER[6], which is the only open-source library that can both download and clean/extract financial data from the United States Securities and Exchange Commission, one of the largest publicly available web repositories with financial and business data. The open-source software is very popular on Github, with a couple of thousands of users, and it has also been used by scientists in the U.S. Federal Reserve (Crane et al., 2024) and the U.S. National Economic Bureau (He et al., 2024).

Next, we list the contributions related to the real-world task of numeric entity recognition and automatic document tagging using XBRL (eXtensible Business Reporting Language, Chapter 3)[7]. XBRL is an XML-based standard that provides a common, machine-readable format for financial data, enabling easier analysis and exchange. It is also required by the US SEC regulations.

We introduce XBRL tagging, a new financial NLP task for a real-world need, and we release FiNER-139, the first XBRL tagging dataset (Chapter 3): FiNER-139 contains 1M sentences with XBRL gold manually expert-assigned labels.

LSTMs operating on word embeddings beat BERT and LLMs in numeric entity recognition (Chapter 3): This finding is correlated with the use of context-sensitive numeric entity names in financial text, as well as the extreme fragmentation of tokens by Transformer-based models. The finding is also impressive, in the sense that BERT and (even more) LLMs are vastly pre-trained on large plain-text corpora, whereas the LSTMs are not pre-trained, except for their word embeddings.

We propose a new, state-of-the-art, tokenization technique for Transformer models, which shows better results in numeric-related tasks (Chapter 3): We demonstrate that substituting numeric tokens with pseudo-tokens encoding their morphological patterns and approximate magnitudes leads to substantial performance improvements in BERT-based models for this task, as well as LLMs. This tokenization strategy proves critical to enabling Transformer models to process and leverage numerical information effectively.

We release a new family of BERT models for the financial domain (Chapter 3): We pre-train a BERT model on 200k financial filings, incorporating our proposed tokenization strategy securing state-of-the-art performance. The models are publicly released and can be

[6] https://github.com/nlpaueb/edgar-crawler/

[7] https://www.xbrl.org/

used in future research as well as financial applications.

In the real-world context of resource-limited intent recognition in banking (Chapter 4), the contributions of the thesis can be summarized as follows.

We conduct a comprehensive benchmarking study of few-shot intent recognition methods on Banking77 (Chapter 4): This includes fine-tuning MPNet (Song et al., 2020), one of the leading sentence transformers models[8], with Full-Data, as well as Few-Shot training via SetFit (Tunstall et al., 2022) and in-context learning with commercial LLMs using 1-3 examples per class.

We show that smaller BERT-based models can outperform LLMs in Few-Shot settings when trained with contrastive learning (Chapter 4): For example, an MPNet-v2 Sentence Transformer trained using SetFit can achieve better performance than GPT-3.5-turbo. Also, with with just a few more labeled examples per class in SetFit (e.g., 5 instead of 3), these smaller models not only perform better than state-of-the-art LLMs like GPT-4 (with 3 examples in context), but they are also more cost-effective to deploy than most LLMs.

We demonstrate that BERT-based models, fine-tuned on a full dataset, significantly outperform general-purpose, powerful and expensive LLMs in intent recognition, a classification task (Chapter 4): When fine-tuned on the complete Banking77 dataset, MPNet-v2 substantially surpasses the performance of all tested large general-purpose language models for this classification task.

We highlight the significant cost-performance advantages of smaller, efficient BERT-based models (Chapter 4): Models like MPNet-v2 (438MB, runnable on CPU) offer a vastly superior cost-to-performance ratio compared to expensive API-based LLMs like GPT-4 for such classification tasks.

We show that using expert-curated examples for in-context learning yields substantial performance gains (Chapter 4): Compared to randomly selected examples, expert-curated samples improve LLM performance by up to 10 points, a practical gain as obtaining a few high-quality samples per class is often feasible.

We investigate and demonstrate the cost-effectiveness of "Dynamic Few-Shot Prompting" for commercial LLM APIs (Chapter 4): Utilizing Retrieval-Augmented Generation (RAG) to dynamically select the most similar training samples during inference, this approach significantly cuts API costs while maintaining or improving performance. For example, GPT-4 with dynamic prompting (5 retrieved examples) reduces costs by 72% compared to standard

[8] `https://www.sbert.net/docs/sentence_transformer/pretrained_models.html`

3-shot prompting ($205 vs. $740 on the Banking77 test set) while improving accuracy from 83.1% to 84.5%. By retrieving only a small but effective fraction (2.2%) of available examples, this method drastically reduces input token usage, demonstrating that targeted example selection is both more cost-efficient and achieves better performance than including all examples in the prompt as done in standard few-shot prompting. To the best of our knowledge, this is the first study that comprehensively evaluates the cost-performance trade-offs of few-shot methods for commercial LLM APIs in a resource-limited industrial context, where both data availability and budget constraints are key factors.

We investigate replacing expert-curated samples with GPT-4 synthetic data augmentation (Chapter 4): In low-resource simulations, this improves performance over minimal expert data, reducing manual annotation needs, though real expert-curated data remains superior.

1.4 Outline of the remainder of this thesis

The remainder of this thesis is organized as follows.

Chapter 2 presents our work on developing open-source software and resources for financial NLP. We detail the creation of a large-scale corpus, domain-specific word embeddings, and a toolkit to make financial documents more accessible and used for NLP applications.

Chapter 3 focuses on Numeric Entity Recognition for business documents. We introduce the real-world task of XBRL Tagging, a regulatory requirement from the U.S. SEC, and propose new tokenization methods to effectively handle the numeric nature of financial text, leading to state-of-the-art performance.

Chapter 4 addresses resource-limited intent recognition in banking. This chapter provides a comprehensive benchmarking study of few-shot methods and analyzes the cost-performance trade-offs for the deployment of machine learning models. We also explore the benefits of expert-curated data and effective strategies like Retrieval-Augmented Generation for querying Large Language Models.

Finally, Chapter 5 summarizes the key findings of this thesis and discusses potential directions for future research.

Chapter 2

Open-Source Software and Resources

2.1 Introduction

While financial data often appear in structured tables, significant valuable information resides within textual sources, as documented by related surveys (Fisher et al., 2016; Loughran and McDonald, 2016; El-Haj et al., 2019; Loughran and McDonald, 2020). Extracting insights from this text is crucial for a deeper understanding of market dynamics and company performance. A primary, freely accessible repository for such textual data is the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system, maintained by the US Securities and Exchange (SEC).[1] Since its inception in 1993, EDGAR has become one of the most comprehensive global archives of financial information, hosting mandatory filings from publicly traded US-based companies. These filings, with around 3,000 of them uploaded daily,[2] aim to ensure market transparency by detailing companies' financial status and significant events like acquisitions or bankruptcies (Kogan et al., 2009).

The documents available on EDGAR are an invaluable resource for various machine learning (ML) and NLP applications. Researchers have utilized them for tasks including stock price prediction (Nassirtoussi et al., 2014; Xing et al., 2017; Lee et al., 2014), sentiment analysis (Kearney and Liu, 2014), risk analysis (Kogan et al., 2009), IPO (initial public offering) underpricing (Katsafados et al., 2023a), financial distress or bankruptcy prediction (Gandhi et al., 2019; Gkoumas et al., 2024), and identifying merger participants (Katsafados et al., 2023b). Often, research and applications utilizing those documents focuses on specific filing types, years, or companies. Among the most frequently used filings are:

[1] See `https://www.sec.gov/edgar/searchedgar/companysearch.html` for more information.

[2] `https://en.wikipedia.org/wiki/EDGAR`

- (10-K) annual reports, offering comprehensive overviews including audited statements and risk factors (Kogan et al., 2009; Griffin, 2003; Zavitsanos et al., 2022; Loukas et al., 2021c, 2022; Sharma et al., 2023);
- (10-Q) quarterly reports, providing interim financial updates (Griffin, 2003; Lee, 2012; Balsam et al., 2002);
- (8-K) current reports, disclosing important corporate events (Lee et al., 2014; Carter and Soo, 1999) like mergers and acquisitions or equity dilutions.

To use such financial data from EDGAR in their research, NLP practitioners often rely on (paid) curated-dataset providers. Examples include Wharton Research Data Services (WRDS)[3] or NASDAQ Premium Data Publishers[4], which require costly institutional licenses, often costing tens of thousands of dollars per year, which are prohibitive for smaller research groups or budget-limited organizations. More affordable, yet still paywalled, options also exist like AWS Marketplace[5]. Free alternatives exist, but are limited to simply downloading the raw web documents from EDGAR without curating them (filtering or structuring them), which presents a significant challenge when actually trying to use them in an NLP workflow.

Metric	Form 10-K	Form 10-Q	Form 8-K
Median Amount of Pages	115 pages	45 pages	3 pages
Range of Pages	50–300+	20–100+	1–10+ (depending on event)
Median Word Count	~46,000 words	~23,500 words	~450 words
Range of Word Count	23,000–70,000+	10,000–40,000+	100–5,000+
Main Parts / Sections	4 parts / 21 required items	2 parts / 11 required items	No parts; up to 32 event items
Typical Use	Annual performance, risks, strategy	Quarterly financial updates	Real-time material events

Table 2.1: Descriptive statistics for the various EDGAR filings: 10-Ks (annual reports), 10-Qs (quarterly reports), and 8-Ks (current reports), which have also been the focus of prior studies (Cazier and Pfeiffer, 2016; Wallek and Ebert, 2016; Loughran and McDonald, 2016)

For instance, the raw financial documents on EDGAR can have different formats depending on the date of origin (.txt or .html), are notoriously complex and lengthy – 10-K reports, for example, average 55,000 words and are divided into (up to) 23 main item sections (Cazier and Pfeiffer, 2016), while 10-Q and 8-K reports contain 11 and 32 sections, respectively (Table 2.1 and 2.1). To strengthen this argument, for example, Dyer et al. (2017) has found that the

[3] https://wrds-www.wharton.upenn.edu/pages/about/data-vendors/

[4] https://data.nasdaq.com/publishers

[5] https://aws.amazon.com/marketplace/pp/prodview-k65p3l3pdyqfw#offers

median length of a 10-K report has also doubled through years 1996 to 2013, while readability and specificity declined, indicating that the root causes come from regulatory requirements. Additionally, these regulatory documents incorporate both tabular and structured data (e.g., balance sheets and financial statements; see Table 2.2) as well as textual commentary (text notes) that provide context and analysis (Table 2.3). Thus, this complexity and the sheer document size, which cannot fit in the context of most large language models, make manual processing impractical and automated processing difficult.

TABLE OF CONTENTS

		Page
	PART I	
Item 1.	Business	1
Item 1A.	Risk Factors	7
Item 1B.	Unresolved Staff Comments	17
Item 1C.	Cybersecurity	17
Item 2.	Properties	19
Item 3.	Legal Proceedings	19
Item 4.	Mine Safety Disclosures	19
	PART II	
Item 5.	Market for Registrant's Common Equity, Related Stockholder Matters and Issuer Purchases of Equity Securities	20
Item 6.	Reserved	21
Item 7.	Management's Discussion and Analysis of Financial Condition and Results of Operations	22
Item 7A.	Quantitative and Qualitative Disclosures About Market Risk	30
Item 8.	Financial Statements and Supplementary Data	31
Item 9.	Changes in and Disagreements with Accountants on Accounting and Financial Disclosure	62
Item 9A.	Controls and Procedures	62
Item 9B.	Other Information	65
Item 9C.	Disclosure Regarding Foreign Jurisdictions That Prevent Inspections	65
	PART III	
Item 10.	Directors, Executive Officers and Corporate Governance	66
Item 11.	Executive Compensation	66
Item 12.	Security Ownership of Certain Beneficial Owners and Management and Related Stockholder Matters	66
Item 13.	Certain Relationships and Related Transactions, and Director Independence	66
Item 14.	Principal Accountant Fees and Services	66
	PART IV	
Item 15.	Exhibits and Financial Statement Schedule	67
Item 16.	Form 10-K Summary	69
SIGNATURES		70

Figure 2.1: The table of contents from a randomly sampled 10-K document. The original document can be found at `https://www.sec.gov/Archives/edgar/data/1326380/000162828025014731/gme-20250201.htm`

.

Moreover, in practice, researchers typically focus on only a few specific sections relevant to their experiments (Table 2.5) instead of utilizing the whole document.[6] For example, Katsafados et al. (2023a) combine just Item 7 and Item 1A to detect Initial Public Offering underpricing while Goel and Uzuner (2016) use Item 7 only to analyze the management's sentiment for fraud detection. Finding the specific information one wants to use for their research in SEC filings is a main challenge if one does not have access to proprietary data from

[6] `www.investopedia.com/articles/basics/10/efficiently-read-annual-report.asp`

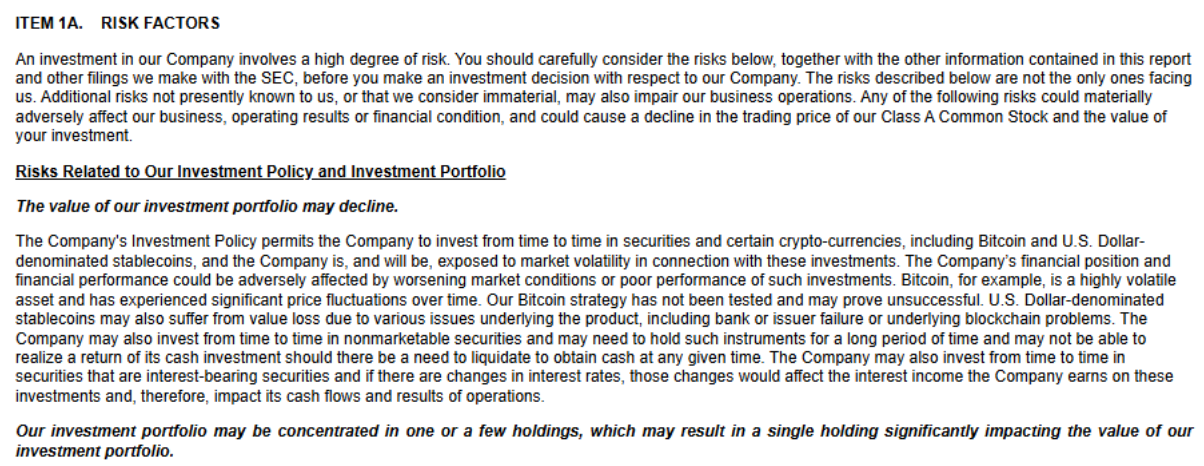
paid services or licensed dataset distributors like Bloomberg.

GAMESTOP CORP.
CONSOLIDATED BALANCE SHEETS
(in millions, except par value per share)

	February 1, 2025	February 3, 2024
ASSETS		
Current assets:		
Cash and cash equivalents	$ 4,756.9	$ 921.7
Marketable securities	18.0	277.6
Receivables, net of allowance of $4.7 and $4.4, respectively	60.9	91.0
Merchandise inventories, net	480.2	632.5
Prepaid expenses and other current assets	39.0	51.4
Total current assets	5,355.0	1,974.2
Property and equipment, net of accumulated depreciation of $684.2 and $851.2, respectively	68.2	94.9
Operating lease right-of-use assets	374.1	555.8
Deferred income taxes	18.1	17.3
Other noncurrent assets	60.0	66.8
Total assets	$ 5,875.4	$ 2,709.0
LIABILITIES AND STOCKHOLDERS' EQUITY		
Current liabilities:		
Accounts payable	$ 148.6	$ 324.0
Accrued liabilities and other current liabilities	362.2	412.0
Current portion of operating lease liabilities	144.3	187.7
Current portion of long-term debt	10.3	10.8
Total current liabilities	665.4	934.5
Long-term debt	6.6	17.7
Operating lease liabilities	249.5	386.6
Other long-term liabilities	24.1	31.6
Total liabilities	945.6	1,370.4
Stockholders' equity:		
Class A Common Stock — $.001 par value; authorized 1,000 shares; 447.0 and 305.7 shares issued and outstanding, respectively	0.2	0.1
Additional paid-in capital	5,105.1	1,634.9
Accumulated other comprehensive loss	(94.0)	(83.6)
Retained loss	(81.5)	(212.8)
Total stockholders' equity	4,929.8	1,338.6
Total liabilities and stockholders' equity	$ 5,875.4	$ 2,709.0

See accompanying notes to consolidated financial statements.

Figure 2.2: The consolidated balance sheet of GameStop ($GME) for year 2024, as found in their original 10-K document (annual report). Retrieved from `https://www.sec.gov/Archives/edgar/data/1326380/000162828025014731/gme-20250201.htm`. This part comes from Item 8 of the document.



ITEM 1A. RISK FACTORS

An investment in our Company involves a high degree of risk. You should carefully consider the risks below, together with the other information contained in this report and other filings we make with the SEC, before you make an investment decision with respect to our Company. The risks described below are not the only ones facing us. Additional risks not presently known to us, or that we consider immaterial, may also impair our business operations. Any of the following risks could materially adversely affect our business, operating results or financial condition, and could cause a decline in the trading price of our Class A Common Stock and the value of your investment.

Risks Related to Our Investment Policy and Investment Portfolio

The value of our investment portfolio may decline.

The Company's Investment Policy permits the Company to invest from time to time in securities and certain crypto-currencies, including Bitcoin and U.S. Dollar-denominated stablecoins, and the Company is, and will be, exposed to market volatility in connection with these investments. The Company's financial position and financial performance could be adversely affected by worsening market conditions or poor performance of such investments. Bitcoin, for example, is a highly volatile asset and has experienced significant price fluctuations over time. Our Bitcoin strategy has not been tested and may prove unsuccessful. U.S. Dollar-denominated stablecoins may also suffer from value loss due to various issues underlying the product, including bank or issuer failure or underlying blockchain problems. The Company may also invest from time to time in nonmarketable securities and may need to hold such instruments for a long period of time and may not be able to realize a return of its cash investment should there be a need to liquidate to obtain cash at any given time. The Company may also invest from time to time in securities that are interest-bearing securities and if there are changes in interest rates, those changes would affect the interest income the Company earns on these investments and, therefore, impact its cash flows and results of operations.

Our investment portfolio may be concentrated in one or a few holdings, which may result in a single holding significantly impacting the value of our investment portfolio.

Figure 2.3: Apart from tables and figures, financial documents also come with lots of text notes.

Thus, the lack of open-source, efficient tools for retrieving and structuring this textual information forces time-consuming manual extraction or reliance on expensive services, slowing down the progress in financial NLP.

2.2 Contributions

Followingly, we present our contributions aimed at overcoming the above-mentioned challenges.

- First, we introduce EDGAR-CRAWLER,[7] the first open-source Python toolkit designed to both download and parse raw financial reports from EDGAR. EDGAR-CRAWLER automates the previously manual and complex steps of data acquisition and curation. EDGAR-CRAWLER converts unstructured web documents (.txt or .html) into a developer-friendly JSON format where all sections and subsections (items) are clearly segmented (Fig. 2.7). EDGAR-CRAWLER offers configuration options for targeting specific companies, years, or filing types, enabling the rapid creation of tailored datasets and empowering researchers, including non-NLP-experts, to leverage EDGAR data without relying on costly services. EDGAR-CRAWLER has been used by multiple researchers and organizations, like scientists from the U.S. Federal Reserve (Crane et al., 2024) and the U.S. Bureau of Economic Analysis (BEA) (He et al., 2024).

[7] EDGAR-CRAWLER is available at: `https://github.com/nlpaueb/edgar-crawler`

- Second, utilizing this toolkit, we constructed and released EDGAR-CORPUS,[8] a large-scale financial corpus comprising all US annual reports (10-K filings) from 1993 to 2020. Each report in the corpus is provided in the structured JSON format generated by EDGAR-CRAWLER, containing all item sections, making the corpus ready-to-use in NLP pipelines for research and/or applications. As shown in Table 2.3, EDGAR-CORPUS represents the largest generally available financial NLP corpus in English. The only larger financial NLP corpus is designed solely for the Chinese language (Lu et al., 2023), while Bloomberg has also constructed an even larger English-language corpus called "FinPile" for pre-training BloombergGPT, but it is proprietary (Wu et al., 2023).

- Finally, leveraging the scale of EDGAR-CORPUS, we trained and released domain-specific (static) word embeddings using WORD2VEC Mikolov et al. (2013a,b), called EDGAR-W2V. Our experimental results demonstrate their superior performance on downstream financial NLP tasks compared to generic embeddings like GloVe (Pennington et al., 2014a) and previous financial embeddings (Tsai et al., 2016), as well as modern LLM-derived embeddings from JinaAI (Günther et al., 2025)[9].

2.3 Related Work

Research using financial text data from the SEC EDGAR system often requires specific functionalities. Researchers' needs typically include: (a) batch crawling documents across multiple years or companies, (b) filtering downloads by identifiers like stock tickers or CIKs (Central Index Key), and crucially (c) mining and extracting clean text from lengthy, unstructured documents, which is the most challenging task (Guo et al., 2016). This last point is particularly important as researchers often need only specific sections from documents, with documents spanning hundreds of pages.

Several open-source toolkits (Table 2.2) have been developed to download financial filings from EDGAR, with varying degrees of success. However, the most common limitation is the lack of support for parsing the downloaded web documents in a clean format, such as splitting them into specific item sections without noise. We briefly discuss existing tools and services below.

sec-edgar-downloader[10] allows users to crawl data based on the company's stock ticker and

[8] EDGAR-CORPUS is available at: `https://zenodo.org/record/5528490`

[9] `https://huggingface.co/jinaai/jina-embeddings-v4`

[10] `https://sec-edgar-downloader.readthedocs.io/en/latest/`

Name	Batch Crawling	Filtered Crawling	Extract Sections	Pricing
sec-edgar-downloader	✗	✓✓	✗	Free
sec-edgar	✓	✓✓	✗	Free
edgartools	✓	✓✓	✗	Free
py-edgar	✗	✓	✗	Free
sec-api.io *(Paid web service)*	✓	✓✓	✓	Subscription Paywall
Snowflake *(Paid data product)*	✓	✓✓	✓	Subscription Paywall
EDGAR-CRAWLER (ours)	✓	✓✓	✓	Free

Table 2.2: EDGAR-related mining OSS libraries, (closed-source) paid web services and data products, along with their functionalities and costs. ✓ denotes minimal support of the feature, while ✓✓ denotes extensive support.

its CIK. However, it cannot clean or extract any data from the documents, making it less useful for NLP.

sec-edgar[11], as well as **edgartools**[12], allow users to crawl data from EDGAR in batches and support filtering. However, they do not support item section extraction, and users must write their own extraction code to integrate the data into their NLP pipelines.

py-edgar[13] allows users to crawl data based on CIKs and does not support the search of companies via stock tickers. It also lacks the ability to parse and extract any data from the documents.

sec-api[14] is a **closed-source** paid service offering batch and filtered crawling. It also recently introduced the ability to parse the documents and produce clean data. However, as a paid service, it limits its usability by small business entities or research groups with limited budgets.

Snowflake's SEC Filings[15] is a **paid data product** offering on the Snowflake Data Marketplace. It provides curated SEC filings, including raw text, parsed XBRL, HTML, and specially sectioned 10-K/10-Q filings for direct querying within the Snowflake cloud platform. Designed for analytics and integration, it functions as a **static, frequently updated resource** queried via SQL, not as a live request-driven API. The wide-format tables support granular analytics (e.g., item-level extraction for LLMs), but access requires a Snowflake account and a commercial agreement.

[11] https://sec-edgar.github.io/sec-edgar/
[12] https://github.com/dgunning/edgartools
[13] https://github.com/joeyism/py-edgar
[14] https://sec-api.io/
[15] https://data-docs.snowflake.com/foundations/products/sec-filings/

The limitations of these existing libraries and services (related to functionality or cost) highlight a gap for an open-source tool that effectively combines comprehensive crawling capabilities (batch and filtered) with automated, structured text extraction from EDGAR documents.

Given the challenges in acquiring and processing raw EDGAR data directly using available tools, researchers have also turned to using already-existing textual financial resources. However, the number of such publicly available resources in the NLP literature is limited. Händschke et al. (2018) published JOCo, a corpus of non-SEC annual and social responsibility reports for the top 270 US, UK, and German companies. Daudert and Ahmadi (2019) released CoFiF, the first financial corpus in the French language, comprising annual, semestrial, trimestrial, and reference business documents.

Other work has focused solely on publishing document collections derived from EDGAR but these often come with certain limitations regarding scope or content. Kogan et al. (2009) published a collection of the Management's Discussion and Analysis Sections (Item 7) for all SEC company annual reports from 1996 to 2006. Tsai et al. (2016) updated that collection to include reports up to 2013 while also providing WORD2VEC embeddings (Mikolov et al., 2013a,b) trained on that data. Focusing on a different filing type, Lee et al. (2014) released a collection of 8-K reports (which announce significant firm events) from 2002 until 2012.

Corpora	Filings	Tokens	Companies	Years	Language
Händschke et al. (2018)	Various	242M	270	2000–2015	English, German
Daudert and Ahmadi (2019)	Various	188M	60	1995–2018	French
Lee et al. (2014)	8-K	27.9M	500	2002–2012	English
Kogan et al. (2009)	10-K	247.7M	10,492	1996–2006	English
Tsai et al. (2016)	10-K	359M	7,341	1996–2013	English
EDGAR-CORPUS (ours)	10-K	**6.5B**	**38,009**	**1993–2020**	**English**

Table 2.3: Financial corpora derived from SEC (lower part) and other sources (upper part). The only works focusing on annual reports (10-K filings) are the ones from Kogan et al. (2009) and Tsai et al. (2016), where they publish only one item (Item 7) from the reports, while in our work (EDGAR-CORPUS), we collect, parse, and publish all item sections.

These previous EDGAR collections are valuable but are often restricted to specific sections (like Item 7) or specific filing types (like 8-K), and cover limited time periods (Table 2.3). This indicates a need for a larger, more comprehensive financial corpus based on EDGAR

filings, specifically one containing the full content (all items) of widely used reports like the 10-K, across a broader range of companies and years, to facilitate diverse research directions in financial NLP (Loughran and McDonald, 2016). Additionally, without having a toolkit to parse and use new data, research will always be limited to recycling the same existing datasets, which constrains innovation and comprehensive analysis.

2.4 EDGAR-CRAWLER

As introduced above, the first contribution of this thesis is EDGAR-CRAWLER[16], the only open-source Python toolkit designed to both download and parse raw financial reports from EDGAR. EDGAR-CRAWLER automates the previously manual and complex steps of data acquisition and curation, by converting unstructured financial web documents (.txt or .html) into a researcher-friendly JSON format.

2.4.1 Software Architecture

EDGAR-CRAWLER consists of two Python modules:

`download_filings.py` is responsible for crawling and downloading financial reports for publicly traded companies. It supports multiple input arguments provided by the user, such as the start and end year, quarters, filing types, and a list of company identifiers to download reports for. We utilize `BeautifulSoup4`[17] for the crawling and we set a maximum rate of 10 HTTP GET requests per second, as per EDGAR's website policy.

`extract_items.py` cleans and extracts the text of all or particular items from downloaded documents, regardless of file format (e.g., older .txt or modern .html filings). Users can select which sections to extract and optionally remove numerical tables. Special characters, extra spaces/newlines, HTML tags etc. are automatically removed to obtain the clean text data. The extracted data is saved as JSON files, with each item's text stored as a separate key-value pair (Fig. 2.7), for easy integration in NLP pipelines.[18]

The parsing mechanism of the extraction module uses a pre-defined mapping of item section headers, such as `"Item 1."`, `"Item 1A."`, and so on, to mark the beginning of each

[16] `https://github.com/nlpaueb/edgar-crawler`

[17] `https://beautiful-soup-4.readthedocs.io/en/latest`)

[18] Multi-threaded extraction is powered by the `pathos` library for efficiency (`https://pypi.org/project/pathos/`).

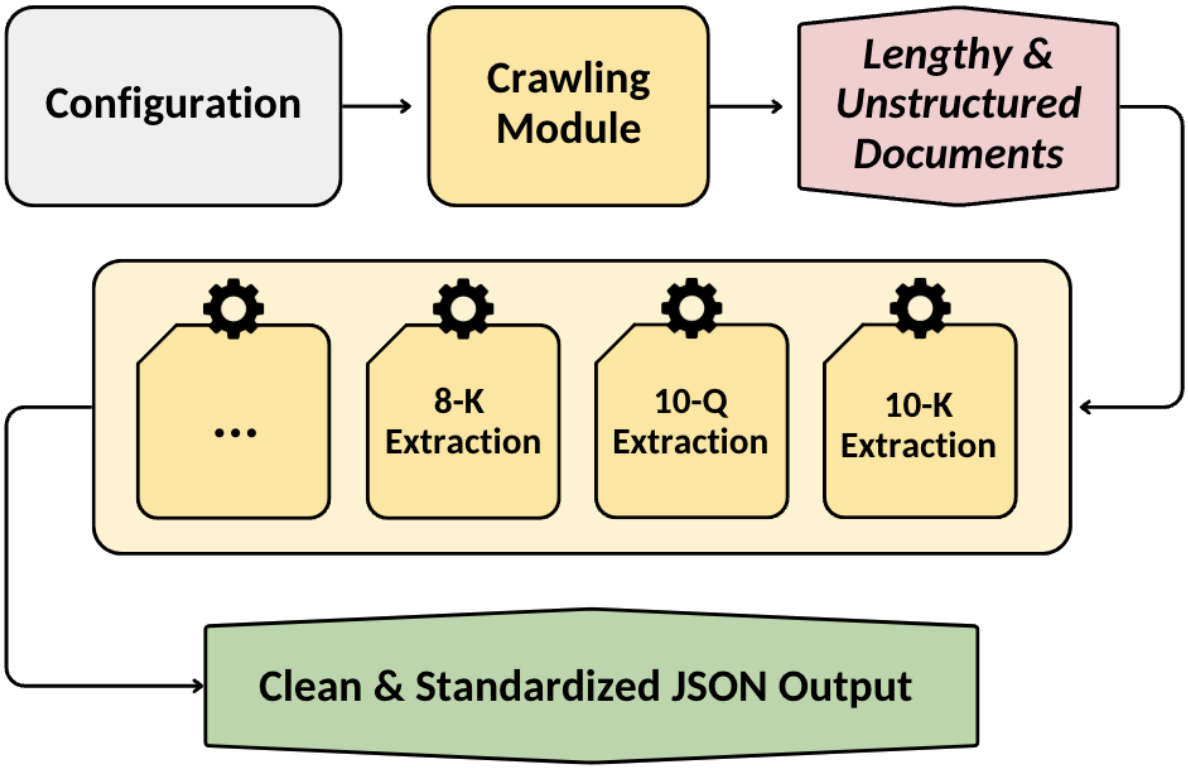


Figure 2.4: EDGAR-CRAWLER's architecture. First, the user specifies (Configuration) what data they want (companies, years, filing types) and the software downloads them. Then, the item extraction pipeline extracts the item-specific sections and converts them to a standardized JSON format.

relevant section. The parser iterates through the document line-by-line, registering the character offset or line index where each section header first appears. To identify the boundaries of each section, the parser captures the text starting from each detected header up until the next identified header in the sequence. This approach ensures that only the main body of each item is extracted, while avoiding common noise such as repeated headers that may appear in footers or exhibit attachments.

The header detection itself relies on a combination of regular expressions and positional tracking. Regular expressions like `r'Item 1A.'` (case-insensitive) are used to match section headers reliably. These regular expressions were meticulously created over many iterations of manually analyzing company filings over various years.

During a first pass, the parser records the position of each unique item header's first occurrence. If the same header appears again later in the document, something which happens frequently due to the table of contents as well as references, the parser flags it as a duplicate based on its distance from the initial occurrence and its context within the document structure. These duplicates are either ignored or used to help define section boundaries more accurately (see the example below). The result is a clean extraction where each item section, such as `Item 1A. Risk Factors`, is captured only once and without contamination from repeated text elsewhere in the document.

Since a specific item can also be mentioned inside paragraphs of other items (without being an actual item section header), we apply several precautions to avoid cutting an item section short or mistakenly detecting an item header in the wrong place. To ensure this, we require that each item header is explicitly preceded by a newline character (`\n`), reducing the likelihood of matching inline references. We also perform a case-sensitive search first, resorting to a lowercase-insensitive match only as a fallback if no match is found. In the rare cases where multiple candidate matches exist, we retain the longest detected span to maximize the chance of capturing the complete section text. The complete list of the item headers can be found at Table 2.5.

Example of the parsing mechanism. We show a typical example of this parsing mechanism (for a 10-K report) below. Let's imagine that we have the following example document:

```
... some intro text ...
ITEM 1. Business
The Company does X, Y, Z...
```

```
[some paragraphs]
ITEM 1A. Risk Factors

Risk 1: exposure to ...

Risk 2: competition ...
[some paragraphs]
ITEM 1B. Unresolved Staff Comments
None.
... [other items] ...
... later in exhibits ...
ITEM 1A. Risk Factors (continued)
```

In this case, `edgar-crawler` will:

- Identify the first occurrence of `"ITEM 1. Business"` and mark it as the start of `item_1`.
- Detect `"ITEM 1A. Risk Factors"` and mark it as the start of `item_1A`.
- Detect `"ITEM 1B. Unresolved Staff Comments"` as the start of `item_1B`.
- Extract `item_1` content from `"ITEM 1. Business"` up to `"ITEM 1A."`, `item_1A` from `"ITEM 1A."` up to `"ITEM 1B."`, and so forth.
- Ignore the later repeated header `"ITEM 1A. Risk Factors (continued)"` since it occurs after the first captured boundaries and within less relevant parts like exhibits.

The same logic applies in other filings like 8-K (current reports) and 10-Q (quarterly reports) filings. Also, since 10-Q filings are separated into two parts (one with financial data and one with non-financial but important data) whose items can have the same names, we follow the above approach, but hierarchically: we first identify the two parts, and then search each part separately for its item sections. The source code of the toolkit can be found also in Appendix A.

2.4.2 Configuration and Further Examples

Before running any of the Python modules, one needs to configure the toolkit through the `config.json` file. This configuration file specifies parameters such as filing types, date ranges, company identifiers, output formats, and more. Followingly (Figure 2.5 and Figure 2.6), we provide two example scenarios of how to use EDGAR-CRAWLER: one for single-company queries and another for batch processing across multiple companies. For up-to-date documentation and additional examples, please consult our GitHub.[19]

```
{"download_filings": {
    "start_year": 2014,
    "end_year": 2020,
    "quarters": [1, 2, 3, 4],
    "filing_types": ["10-K"],
    "cik_tickers": ["MSFT"],
    "user_agent": "your_name@mail.com",
    "raw_filings_folder": "RAW_FILINGS",
    "indices_folder": "INDICES",
    "filings_metadata_file": "FILINGS_METADATA.csv",
    "skip_present_indices": true},
"extract_items": {
    "raw_filings_folder": "RAW_FILINGS",
    "extracted_filings_folder": "EXTRACTED_FILINGS",
    "filings_metadata_file": "FILINGS_METADATA.csv",
    "items_to_extract": ["1", "1A", "1B", "7", "7A", "8"],
    "remove_tables": true,
    "skip_extracted_filings": true}}
```

Figure 2.5: Example configuration for downloading Microsoft's 10-K filings (2014-2020) and extracting items 1, 1A, 1B, 7, 7A, 8. Key parameters: `skip_present_indices` avoids re-downloading master index files, `remove_tables` strips numerical tables, `skip_extracted_filings` skips parsed documents. Finally, the parameters `FILINGS_METADATA.csv` and `EXTRACTED_FILINGS` coordinate the metadata across the two pipeline stages.

[19] `https://github.com/nlpaueb/edgar-crawler/`

```
{"download_filings": {
    "start_year": 2021,
    "end_year": 2021,
    "quarters": [1, 2, 3, 4],
    "filing_types": ["10-K"],
    "cik_tickers": [],
    "user_agent": "Your name (your email)",
    "raw_filings_folder": "RAW_FILINGS",
    "indices_folder": "INDICES",
    "filings_metadata_file": "FILINGS_METADATA.csv",
    "skip_present_indices": true
}, "extract_items": {
    "raw_filings_folder": "RAW_FILINGS",
    "extracted_filings_folder": "EXTRACTED_FILINGS",
    "filings_metadata_file": "FILINGS_METADATA.csv",
    "items_to_extract": ["1", "1A", "1B", "2", "3", "4",
                         "5", "6", "7", "7A", "8", "9",
                         "9A", "9B", "10", "11", "12","13",
                         "14", "15"],
    "remove_tables": false,
    "skip_extracted_filings": true}}
```

Figure 2.6: Example configuration showing how to download all 10-K filings for year 2021 from all companies, extract all items, and not removing any table.

2.4.3 JSON Output

After running the toolkit, there will be multiple item-segmented JSON files, one for each filing, containing the clean and structured text data. Each JSON file is structured with clearly labeled sections corresponding to the filing's items, making it straightforward to access specific parts of the document programmatically to help in research or use in any downstream task. An example of the JSON output is shown in Figure 2.7.

```
{
  "filename": "92116_10K_2004.html",
  "cik": "92116",
  "year": "2004",
  "..": "...",
  "item_1": "Item 1. Business ...",
  "item_1A": "Item 1A. Risk Factors: ...",
  "item_1B": "Item 1B. Unresolved Staff Comments: None.",
  "...": "...",
  "item_7": "Item 7. Management's Discussion and Analysis: ...",
  "item_7A": "Item 7A. Quantitative and Qualitative Disclosures: ...",
  "....": "...",
  "item_15": "Item 15. Exhibits and Financial Statement Schedules: ...",
  "item_16": "Item 16. Form 10-K Summary: ..."
}
```


Figure 2.7: Example of a financial filing downloaded and parsed to a JSON format by EDGAR-CRAWLER. The original filing is a 100-page unstructed .html file (`https://www.sec.gov/Archives/edgar/data/92116/000095015004000378/a97331e10vk.htm`. EDGAR-CRAWLER supports multiple EDGAR filing types (10-K, 10-Q, 8-K).

2.4.4 Impact in other NLP and AI work

EDGAR-CRAWLER currently has 400+ stars on Github and it is growing steadily (Fig. 2.8). Apart from its adoption by practitioners, it is also worth noting that it has been used in multiple NLP and AI studies. For example, the U.S. National Bureau of Economic Research used EDGAR-CRAWLER to research customer capital (He et al., 2024), while the U.S. Federal Reserve Board tracked specific 8-K item sections to predict layoffs (Crane et al., 2024). Researchers from Goldman Sachs performed business XML tagging by leveraging custom BERT models (Loukas et al., 2022) pre-trained on data generated by our software (Sharma et al., 2023). Moreover, EDGAR-CRAWLER has enabled the creation of several financial datasets, including datasets for specific tasks like fraud detection (Zavitsanos et al., 2022).

In addition, EDGAR-CRAWLER has contributed in advancing NLP research benchmarks. For instance, Cao et al. (2024) benchmarked LLMs from OpenAI and Anthropic on long-form document summarization using data from EDGAR-CRAWLER, while Hillebrand et al. (2022) utilized Word2vec embeddings (Mikolov et al., 2013a) trained on data generated by our toolkit to perform relation extraction. Finally, Ahn et al. (2024) developed a graph-based investing method using EDGAR-CRAWLER data, outperforming the average yearly return ratio

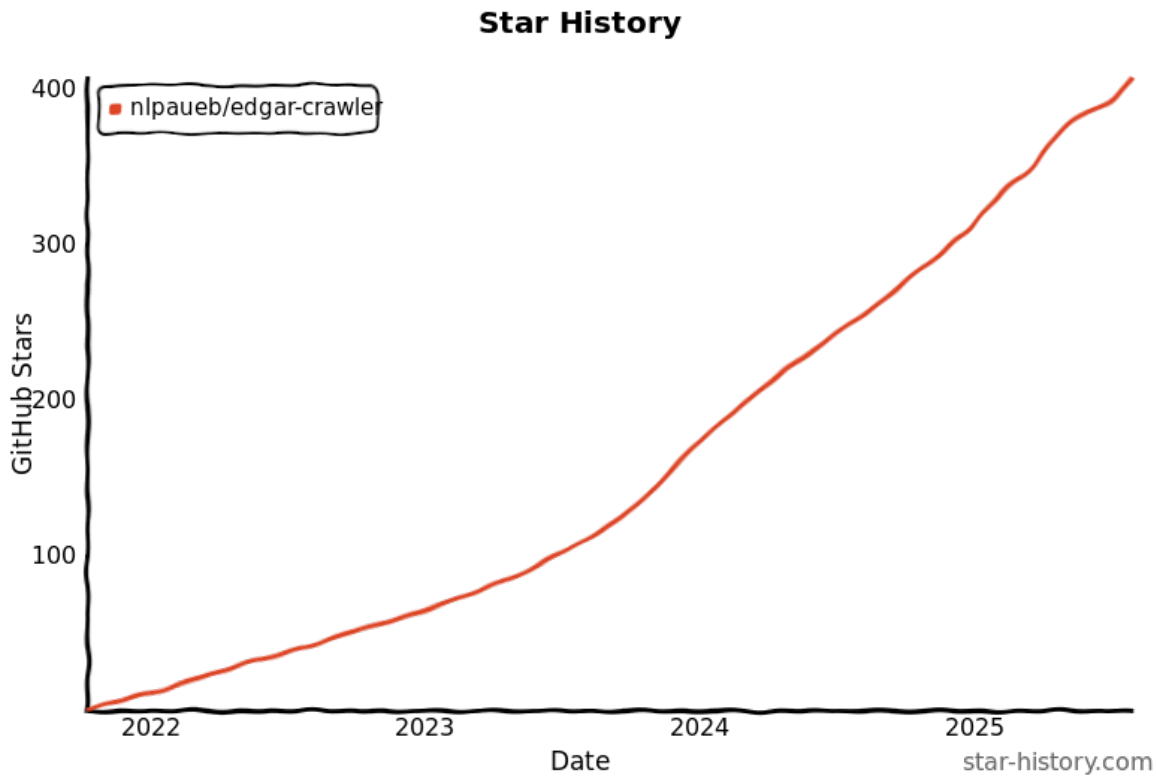


Figure 2.8: Github stars of EDGAR-CRAWLER, across the years, as of July 2025.

of the S&P 500 Index[20].

2.4.5 Parsing Accuracy Tests

Evaluating the parsing accuracy of EDGAR-CRAWLER is challenging due to the absence of a standardized, annotated dataset of financial filings, something that would demand substantial human effort. To address this, we conducted a small-scale manual evaluation to verify the tool's performance on recent documents from 2023 and 2024. Since most of the software was developed between 2019 and 2021, these later documents from 2023 and 2024 serve as a robust test subset to assess the tool's effectiveness on chronologically newer data.

For this test, we selected a sample of 60 companies, comprising thirty random tickers for the 2023 filing year and another thirty for the 2024 filing year. To ensure a representative sample, each set of thirty tickers was stratified to include 10 small-cap (companies with a market capitalization between $300 million and $2 billion), 10 mid-cap ($2 billion to $10 billion), and 10 large-cap (over $10 billion) companies. The specific tickers used are detailed in Table 2.4. We then ran EDGAR-CRAWLER to download and parse the 10-K annual reports

[20] https://en.wikipedia.org/wiki/S%26P_500

for these companies.

Market Cap	2023 Tickers	2024 Tickers
Small-cap	AAP, ABM, ACLS, ACAD, ACIW, ENACT, ABR, ACA, AMRB, TBBK	ALE, BKH, CFR, DX, ELRS, FULT, GMED, HWC, IGMS, JJSF
Mid-cap	AOS, PLNT, ORI, BLD, WBS, CPB, HQY, KVYO, KD, FND	KRTX, LW, MTTR, NATI, ORLY, PWR, RMD, SNA, TTWO, ULTA
Large-cap	MSFT, AAPL, AMZN, GOOG, META, NVDA, AVGO, BRK.B, TSLA, JPM	BA, CRM, DIS, GME, HPE, IBM, KO, LMT, ORCL, PEP

Table 2.4: Tickers used for the manual parsing accuracy evaluation of EDGAR-CRAWLER, categorized by market capitalization for the 2023 and 2024 filing years.

Then, we manually verified the structural integrity of the parsed output against the original HTML filings available on the SEC website. Our verification confirmed that EDGAR-CRAWLER successfully extracted all expected items (e.g., Item 1, Item 1A, Item 7, etc.) for the 2023 filings. For the 2024 filings, most sections were also correctly extracted. However, while inspecting the structure of the 2024 filings, we observed that the schema of the 10-K reports had changed. Specifically, we identified a new section, Item 1C, appearing after Item 1B and Item 1A (when Item 1B was not applicable to the company), which of course caused these JSON parsed items to have more text information than needed, due to the software's already defined output schema. A subsequent investigation revealed that during 2023, the SEC adopted new cybersecurity disclosure rules. These rules mandate the inclusion of a new item, "Item 1C", in all annual reports for fiscal years ending on or after December 15, 2023.[21] Therefore, following the parsing accuracy test, we updated both the parsing logic and the schema in EDGAR-CRAWLER to correctly identify and extract this new item. This highlights a fundamental challenge for rule-based parsers: the structure of regulatory filings evolves over time due to legislative and procedural changes. As a result, maintaining the parsing schema and associated regular expressions is an ongoing task. In the next subsection, we explore how EDGAR-CRAWLER can stay up-to-date over time by incorporating a human-in-the-loop framework, supported by AI coding copilots, to adapt to these changes easily.

[21] See: https://www.saul.com/insights/alert/checklist-key-considerations-upcoming-2023-form-10-k-filings-be-filed-2024

2.4.6 Human-in-the-Loop Maintenance with AI coding assistants

As shown in the previous subsection with the introduction of Item 1C, even relatively minor updates in SEC filing formats can affect the output of a (mostly) rule-based system. Such changes, while rare, may lead to noisier outputs in the JSON schema of the software. For reference, in the last 20 years of SEC's EDGAR system, there have been only five major changes in the 10-K item structure: four new items have been introduced and one has been removed. These kinds of modifications require the parsing rules and expected schema to be updated accordingly, even once in a while.

One way to tackle this semi-automatically is by adopting a human-in-the-loop update workflow, where AI coding assistants help streamline changes when needed, as reported by user's issues. For instance, tools like GitHub Copilot Coding Agent[22] can be assigned directly to a user's issue on GitHub. Once the LLM-powered agent is assigned to an issue, the agent reads the issue description, scans the relevant parts of the codebase, proposes a fix, opens a pull request, and even runs the test suite, all without needing a developer to launch an integrated developer environment (IDE). Alternatively, a developer can invoke an AI coding agent directly inside their IDE, describe the problem and affected files, and receive targeted code suggestions almost immediately by the AI coding assistant.

This workflow has already proven useful in the maintenance of EDGAR-CRAWLER. In one GitHub issue[23], EDGAR-CRAWLER was found to be prematurely truncating Item 1 in a rare case when the text inside the header referenced another section such as "Item 1A Risk Factors." The fix proposed by GitHub Copilot[24] modified the regular expression to ensure that the match for ITEM 1 stops only when the actual section header of ITEM 1A starts on a new line, rather than when it merely appears inline. This adjustment made the regex stricter and reduced overmatching, resolving the issue after manual validation. We show a visual example in Figure 2.9.

In another case, documented in GitHub Issue #35,[25], EDGAR-CRAWLER failed to extract sections correctly when filings combined multiple items under a joint header like "Items 1 and 2.", which seems to be very rare. The issue arose because the original regex pattern only matched the singular form "ITEM", missing plural variants. GitHub Copilot[26] proposed

[22] https://github.com/copilot
[23] https://github.com/nlpaueb/edgar-crawler/issues/20
[24] https://github.com/nlpaueb/edgar-crawler/pull/21
[25] https://github.com/nlpaueb/edgar-crawler/issues/35
[26] https://github.com/nlpaueb/edgar-crawler/pull/36

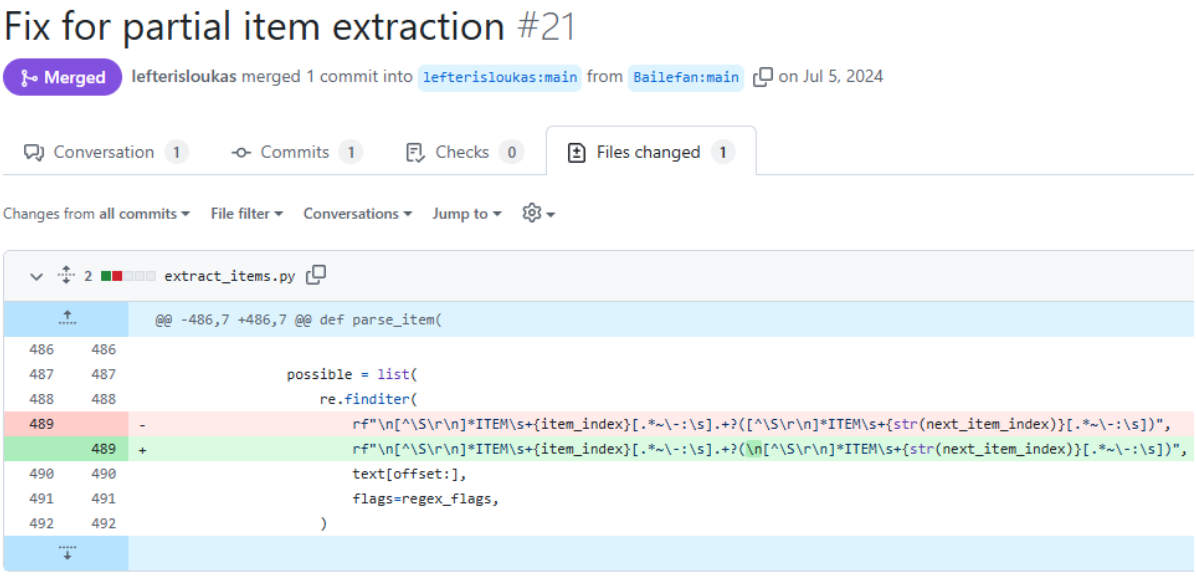


Figure 2.9: A code fix proposed by Github Copilot for issue #20 (`https://github.com/nlpaueb/edgar-crawler/issues/20`), showing the old version in red and the new/proposed version in green, regarding the `parse_item()` function of EDGAR-CRAWLER. The change updates the regex pattern to include an additional `\n` character in the pattern (line 489, green), so that the match ends when a relevant item header is found on newline and not inline.

extending the pattern to accept both "ITEM" and "ITEMS", using a non-intrusive optional character group (Figure 2.10. This minimal change, proposed by an AI assistant, preserved existing matching behavior while expanding coverage of the software, reducing the original time it would take a developer to see the issue, understand it, test different variations, and then actually fix it.

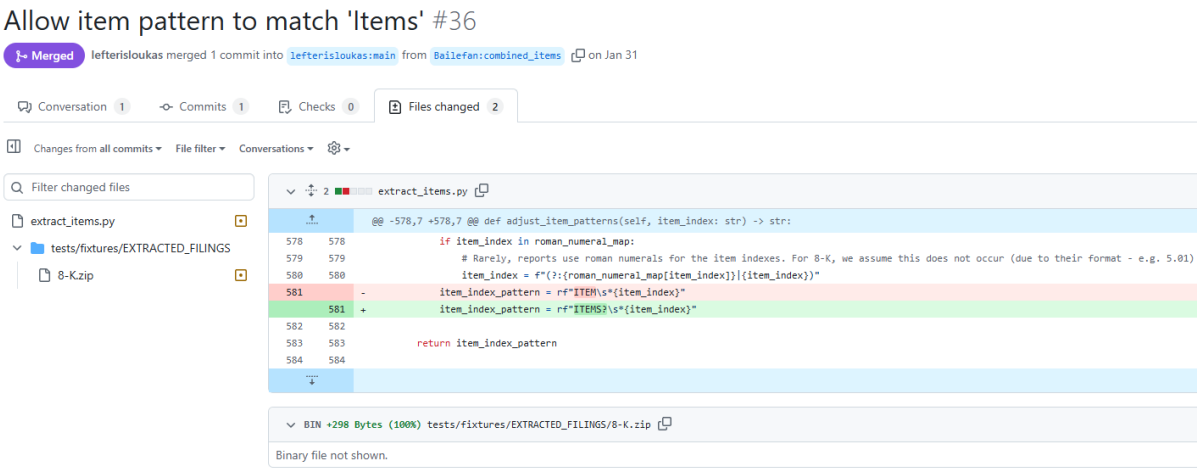


Figure 2.10: A code fix proposed by Github Copilot for issue #35 (`https://github.com/nlpaueb/edgar-crawler/issues/35`), showing the old version in red and the new/proposed version in green, regarding the `parse_item()` function of EDGAR-CRAWLER. The change updates the regex pattern to accept both "ITEM" and "ITEMS" using an optional character group (line 489, green). Apart from the code, we also updated the relevant coverage tests in the Github repository in order to reflect such scenarios, as mentioned in this issue.

2.5 EDGAR-CORPUS

Using the EDGAR-CRAWLER toolkit, we created EDGAR-CORPUS, a static but large-scale NLP resource designed to bootstrap research in the data-limited domain of financial/business NLP.[27] This 6B-tokens corpus addresses the need for comprehensive, structured financial text data by focusing on 10-K filings downloaded and parsed with EDGAR-CRAWLER for over 25 years (1993-2020). These annual filings submitted to the SEC provide a complete description of a company's economic activity, risks, liabilities, corporate agreements, operations, and analysis of the relevant industry sector. Each 10-K report is divided into four parts and contains 20 unique items (Table 2.5); EDGAR-CORPUS provides these items in a pre-parsed collection, overcoming the labor-intensive task of manually retrieving specific sections from lengthy documents or using proprietary services.

[27] EDGAR-CORPUS is available at: `https://huggingface.co/datasets/eloukas/edgar-corpus` and `https://zenodo.org/record/5528490`.

	Item Name	Status
Part I		
Item 1	Business	Required
Item 1A	Risk Factors	Required
Item 1B	Unresolved Staff Comments	If applicable
Item 1C	Cybersecurity	Required
Item 2	Properties	Required
Item 3	Legal Proceedings	Required
Item 4	Mine Safety Disclosures	Mining companies only
Part II		
Item 5	Market	Required
Item 6	Consolidated Financial Data	Reserved
Item 7	Management's Discussion and Analysis	Required
Item 7A	Quantitative and Qualitative Disclosures about Market Risks	Conditional
Item 8	Financial Statements	Required
Item 9	Changes in and Disagreements With Accountants	If applicable
Item 9A	Controls and Procedures	Required
Item 9B	Other Information	If applicable
Item 9C	Disclosure Regarding Foreign Jurisdictions that Prevent Inspections	If applicable
Part III		
Item 10	Directors, Executive Officers and Corporate Governance	Required
Item 11	Executive Compensation	Required
Item 12	Security Ownership of Certain Beneficial Owners	Required
Item 13	Certain Relationships and Related Transactions	Required
Item 14	Principal Accounting Fees and Services	Required
Part IV		
Item 15	Exhibits and Financial Statement Schedules	Required
Item 16	Form 10-K Summary	Optional

Table 2.5: The structure of a 10-K filing with its 23 different items, grouped by section (Parts I–IV), as well as the requirement of each item per the SEC.

2.5.1 Creation process of EDGAR-CORPUS

Using the EDGAR-CRAWLER toolkit described previously, we downloaded the 10-K reports of all publicly traded companies in the US between the years 1993 and 2020. We then processed these filings by removing tables to keep only the text, which was subsequently cleaned (including HTML stripping) and split into the different items using the toolkit's regular expression-based extraction mechanism. The resulting dataset is EDGAR-CORPUS.

The availability of all 20 items in a structured format within EDGAR-CORPUS facilitates diverse research directions, as researchers often rely on specific items for their analyses. For example, as already noted previously, Goel and Gangolly (2012), Purda and Skillicorn (2015), and Goel and Uzuner (2016) perform textual analysis on Item 7 (Management's Discussion and Analysis) to detect corporate fraud. Katsafados et al. (2023a) combine Item 7 and Item 1A (Risk Factors) to detect Initial Public Offering (IPO) underpricing, while Moriarty et al. (2019) combine Item 1 (Business) with Item 7 to predict mergers and acquisitions.

As shown previously in the Related Work section (Table 2.3), EDGAR-CORPUS is significantly larger than previously released financial corpora derived from EDGAR filings. Additionally, EDGAR-CORPUS, after its release, has been used in the creation process of GPT-4 level LLMs like FinTral (Bhatia et al., 2024).

2.5.2 Word Embeddings (EDGAR-W2V)

To further facilitate financial NLP research, we used the data from EDGAR-CORPUS to train WORD2VEC embeddings (EDGAR-W2V), which can be used for downstream tasks, such as financial text classification or summarization. We used WORD2VEC's skip-gram model (Mikolov et al., 2013a,b) with default parameters as implemented by GENSIM (Řehůřek and Sojka, 2010) to generate 200-dimensional WORD2VEC embeddings for a vocabulary of 100K words. The word tokens are generated using SPACY (Honnibal et al., 2020). We also release EDGAR-W2V.[28]

To illustrate the quality of EDGAR-W2V embeddings, in Figure 2.11 we visualize sampled words from seven different entity types, i.e., *location*, *industry*, *company*, *year*, *month*, *number*, and *financial term*, after applying dimensionality reduction with the UMAP algorithm (McInnes et al., 2018). The financial terms are randomly sampled from the Investopedia Financial Terms Dictionary.[29] In addition, companies and industries are randomly sampled from

[28]The EDGAR-W2V embeddings are available at: `https://zenodo.org/record/5524358`.

[29]`https://www.investopedia.com/financial-term-dictionary-4769738`.

well-known industry sectors and publicly traded stocks. Finally, the words belonging to the remaining entity types are randomly sampled from gazetteers. Figure 2.11 shows that words belonging to the same entity type form clear clusters in the 2-dimensional space indicating that EDGAR-W2V embeddings manage to capture the underlying financial semantics of the vocabulary.

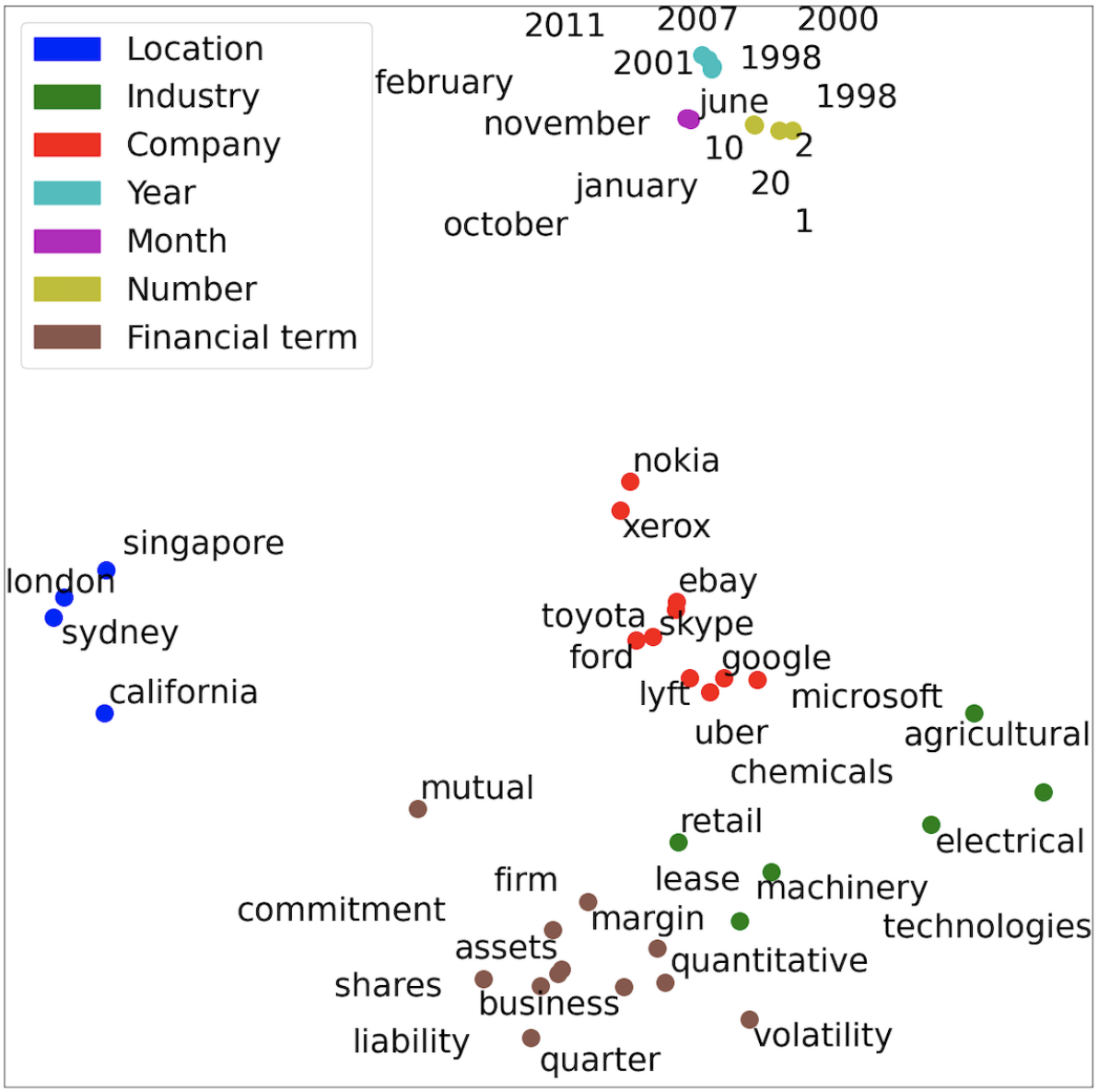


Figure 2.11: Visualization of the EDGAR-W2V embeddings after applying dimensionality reduction with the UMAP algorithm (McInnes et al., 2018). Different colors indicate different entity types.

To further highlight the semantics captured by EDGAR-W2V embeddings, we retrieved the 5 nearest neighbors, according to cosine similarity, for commonly used financial terms

(Table 2.6). We exclude obvious top-scoring neighbors of singular/plural pairs such as *market/markets or investor/investors*. As shown, all the nearest neighbors are highly related to the corresponding term. For instance, the word *economy* is correctly associated with terms indicating the slowdown of the economy, e.g., *downturn*, *recession*, or *slowdown*. Also, *market* is correctly related with words such as *marketplace*, *industry*, and *prices*.

economy	*competitor*	*market*	*national*	*investor*
downturn	competitive	marketplace	association	institutional
recession	competing	industry	regional	shareholder
slowdown	dominant	prices	nationwide	relations
sluggish	advantages	illiquidity	american	purchaser
stagnant	competition	prevailing	zions	creditor

Table 2.6: Sample words from EDGAR-W2V embeddings (top row) and their corresponding nearest neighbors (columns) based on cosine similarity.

2.5.3 Experiments

To assess the effectiveness of EDGAR-W2V embeddings, we compare them against three alternatives: (i) generic GloVe embeddings (Pennington et al., 2014a)[30], (ii) the financial-domain embeddings by Tsai et al. (2016), and (iii) JinaAI's jina-embeddings-v4, a state-of-the-art embedding model based on Qwen2.5-VL-3B-Instruct (Bai et al., 2025), a 3.8 billion parameters model. The evaluation spans three financial NLP tasks, using the same model architecture across all experiments while varying only the embeddings. We also apply identical preprocessing to construct the embedding vocabulary in each case. Our original research (Loukas et al., 2021c), conducted in 2019–2021, preceded the widespread availability of Large Language Models (LLMs). As of the writing of this thesis, however, numerous context-sensitive embeddings derived from LLMs, like jina-embeddings-v4 (which we compare to) are frequently released with strong performance benchmarks (e.g., `https://huggingface.co/spaces/mteb/leaderboard`). These models, though, are substantially larger than static embeddings, making them less practical for local deployment without cloud infrastructure.

[30]We use the 200-dimensional GloVe embeddings from `https://nlp.stanford.edu/data/glove.6B.zip`.

FinSim-3 (Juyeon Kang and Gan, 2021) provides a set of business and economic terms, and the task is to classify them into the most relevant hypernym from a set of 17 possible hypernyms from the Financial Industry Business Ontology (FIBO).[31] Example hypernyms include *Credit Index*, *Bonds*, and *Stocks* (Table 2.7). We tackle the problem with a multinomial logistic regression model, which, given the embedding of an economic term, classifies the term to one of the 17 possible hypernyms. Since the test labels were not available, we use a stratified 10-fold cross-validation. We report the official measures of this task, namely the accuracy and the average rank of the correct hypernym.

Label	**Count**
Equity Index	286
Regulatory Agency	205
Credit Index	129
Central Securities Depository	107
Debt pricing and yields	58
Bonds	55
Swap	36
Stock Corporation	25
Option	24
Funds	22
Future	19
Credit Events	18
Stocks	17
MMIs	17
Parametric schedules	15
Forward	9
Securities restrictions	8
Total	**1050**

Table 2.7: Training set statistics for the FinSim-3 shared task dataset showing the distribution of financial terms across the 17 FIBO ontology categories.

Financial tagging (FiNER) is a sequence labeling problem for financial documents, which

[31] `https://spec.edmcouncil.org/fibo/`.

we actually tackle later in more detail in this thesis (Chapter 3). In high-level, the task is to annotate financial reports with word-level tags from an accounting taxonomy (Loukas et al., 2022). We use a BILSTM encoder operating over word embeddings with a shared multinomial logistic regression that predicts the correct tag for each word from the corresponding BILSTM state. We report the F1 score micro-averaged across all tags.

FiQA Open Challenge (Maia et al., 2018) is a sentiment analysis regression challenge over financial tweets annotated by domain experts with a sentiment score in the range [-1, 1]. We employ a BILSTM encoder operating over word embeddings, and a linear regressor operating over the last hidden state of the BILSTM. We use a 10-fold cross-validation and evaluate using Mean Squared Error (MSE) and R-squared (R^2), which are the official metrics of this task challenge.

Split	Number of Examples	Avg. Sentiment Score	Min Sentiment Score	Max Sentiment Score
Training	822	0.1228	-0.9380	0.9370
Validation	117	0.0305	-0.6260	0.7210
Test	234	0.0202	-0.8120	0.9750

Table 2.8: Statistics for the different splits of the FiQA Sentiment Analysis task.

Across all tasks, EDGAR-W2V outperforms GloVe, showing that in-domain knowledge is critical in financial NLP problems (Table 2.9). The gains are more substantial in FinSim-3 and FiNER, which rely heavily on understanding technical economics discourse. The in-domain embeddings of Tsai et al. (2016), based on the Continuous Bag-of-Words (CBOW) model, a form of WORD2VEC Mikolov et al. (2013a), perform almost the same as the generic GloVe embeddings in two tasks, possibly due to stemming used during their vocabulary creation, which might introduce noise. Lastly, we compare EDGAR-W2V with jina-embeddings-v4 (Günther et al., 2025), a multimodal embedding model with 3.8 billion parameters in the base backbone, based on Qwen2.5-VL-3B-Instruct (Bai et al., 2025), a modern large language model with vision capabilities. Surprisingly, EDGAR-W2V outperforms jina-embeddings-v4 in two of the three tasks: FinSim-3 and FiNER, both of which demand precise handling of domain-specific terms. Additionally, it is worth noting that FinSim-3 consists of short terms with fewer than five tokens per input at inference time, which may limit the capacity of LLM-based embeddings to fully leverage their representational power and the context. In contrast, for sentiment analysis in financial texts (FiQA), jina-embeddings-v4 achieves the highest performance, surpassing EDGAR-W2V and all other baselines by a significant margin,

showcasing its advantage in broader, less specialized financial language tasks.

	FinSim-3		FiNER	FiQA	
	Acc. ↑	Rank ↓	F1 ↑	MSE ↓	R^2 ↑
GloVe	85.3	1.26	75.8	0.151	0.119
Tsai et al. (2016)	84.9	1.27	75.3	0.142	0.169
EDGAR-W2V (ours)	**87.9**	**1.21**	**77.3**	**0.141**	**0.176**
jina-embeddings-v4	83.9	1.32	72.3	**0.110**	**0.302**

Table 2.9: Results across financial NLP tasks, with different static word embeddings, as well as some LLM-derived embeddings. We report averages over 3 runs with different random seeds.

2.6 Conclusions

This chapter addressed the challenge of data scarcity in financial NLP and business documents by focusing on the SEC EDGAR web repository, a rich but often difficult-to-use source of textual information. To improve access to this data, we developed and released EDGAR-CRAWLER. This is the first open-source Python toolkit that combines automated downloading of EDGAR filings with structured parsing, extracting itemized text sections into a clean JSON format and it stands as a free alternative over existing proprietary services.

Using EDGAR-CRAWLER, we then constructed EDGAR-CORPUS, a large-scale, publicly available dataset of parsed 10-K annual reports covering over 25 years (1993-2020). This corpus provides researchers with ready-to-use, structured financial text, removing significant hurdles related to data collection, cleaning, and the cost of proprietary data services.

To demonstrate the utility of EDGAR-CORPUS, we also trained EDGAR-W2V word embeddings on EDGAR-CORPUS. Our experiments showed these domain-specific embeddings outperform standard alternatives as well as modern LLM-derived embedding models (multiple times their size) on three financial NLP tasks.

In conclusion, the EDGAR-CRAWLER toolkit, the EDGAR-CORPUS dataset, and the EDGAR-W2V embeddings offer valuable, practical open-source resources for financial NLP, providing efficient access to SEC EDGAR data and facilitating research and development by leveraging textual information within corporate financial reports. Its modular design allows easy integra-

tion with other machine learning workflows, and ongoing updates ensure compatibility with new SEC filing formats. Together, these resources lower the barrier to entry for financial and business NLP and encourage reproducible research in this domain. EDGAR-CRAWLER is still being maintained after 4 years and has an active community on Github.[32]

[32] https://www.github.com/nlpaueb/edgar-crawler

Chapter 3

Numeric Entity Recognition for XBRL Tagging

3.1 Introduction

After having solved the data scarcity problem for our domain of interest, we focus on how we can utilize such data pertaining to solve a real-world task on business documents, combining methods from Deep Learning and Natural Language Processing. More specifically, we study how financial reports can be automatically enriched with word-level tags from the eXtensive Business Reporting Language (XBRL), a tedious and costly task not considered so far.[1]

Publicly traded corporations are obligated to submit regular financial statements to ensure transparency for both shareholders and prospective investors. These financial reports are uploaded to EDGAR (as described in Chapter 2) and they comprise multiple sections, including financial tables and text paragraphs, called *text notes*. In addition, legislation in the US, the UK, the EU and elsewhere requires the reports to be annotated with tags of XBRL, an XML-based language, to facilitate the processing of financial information. The annotation of tables can be easily achieved by using company-specific pre-tagged table templates, since the structure and contents of the tables in the reports of a particular company rarely change. On the other hand, the unstructured and dynamic nature of text notes (Figure 3.1) makes adding XBRL tags to them much more difficult. Hence, we focus on automatically tagging text notes. Tackling this task could facilitate the annotation of new and old reports (which may not include XBRL tags), e.g., by inspecting automatically suggested tags.

[1] See `https://www.xbrl.org/the-standard/what/an-introduction-to-xbrl/` for an introduction to XBRL.

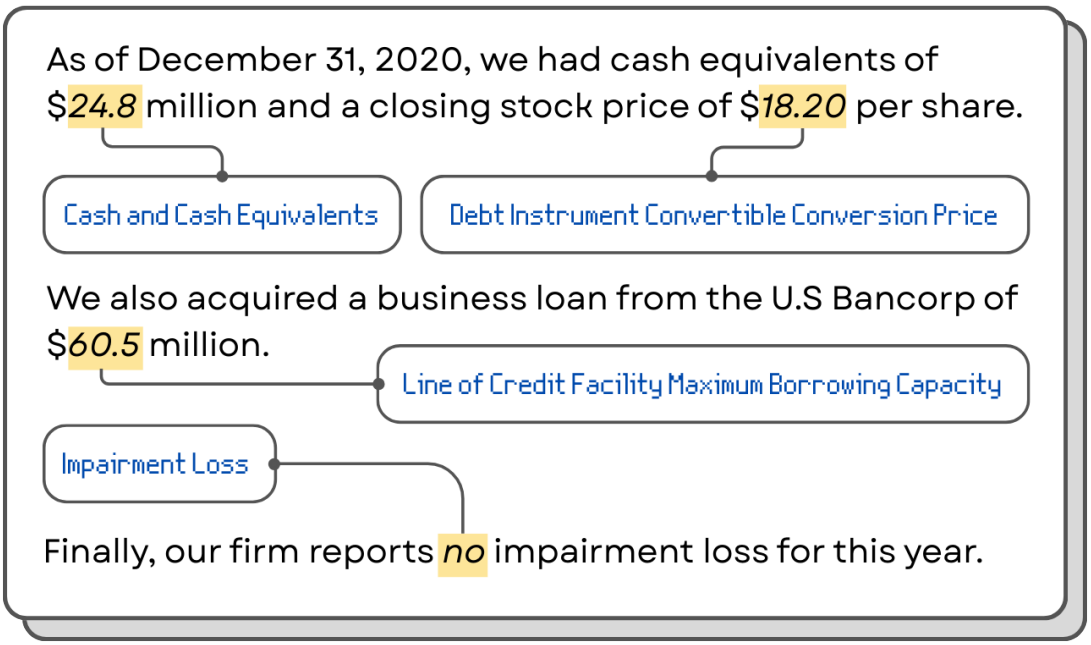


Figure 3.1: Sentences from FiNER-139, the dataset we introduce for XBRL tagging. FiNER-139 contains 1 million sentences with XBRL tags on numeric and non-numeric tokens. XBRL tags are actually XML-based and most tagged tokens are numeric.

For this reason, we release FiNER-139, a new dataset of 1.1M sentences with gold XBRL tags, from annual and quarterly reports of publicly traded companies obtained from the US Securities and Exchange Commission (SEC). Unlike other entity extraction tasks, like named entity recognition (NER) or contract element extraction (Table 3.1), which typically require identifying entities of a small set of common types (e.g., persons, organizations), XBRL defines approximately 6k entity types. As a first step, we consider the 139 most frequent XBRL entity types, still a much larger label set than usual.

A key distinction from standard entity extraction is that most tagged tokens (~91%) in the text notes we consider are numeric, with the correct tag per token depending mostly on context, not the token itself (Figure 3.1). The abundance of numeric tokens also leads to a very high ratio of out-of-vocabulary (OOV) tokens, approx. 10.4% when using a custom Word2Vec (Mikolov et al., 2013a) model trained on our corpus. When using subwords, e.g., in models like BERT (Devlin et al., 2019a), there are no OOV tokens, but numeric expressions get excessively fragmented, making it difficult for the model to gather information from the fragments and correctly tag them all. In our experiments, this is evident by the slightly better performance of stacked BILSTMs (Graves et al., 2013; Lample et al., 2016a) operat-

ing on word embeddings compared to BERT (Section 3.6). The latter improves when using a CRF (Conditional Random Field) layer (Lafferty et al., 2001), which helps avoid assigning nonsensical sequences of labels to the fragments (subwords) of numeric expressions (Section 3.9).

To further improve BERT's performance, we propose two simple and effective solutions that replace numeric expressions with pseudo-tokens reflecting the original token shapes and magnitudes. We also experiment with FIN-BERT (Yang et al., 2020), an existing BERT model for the financial domain, and release our own family of BERT models, pre-trained on 200k financial filings, achieving the best overall performance.

3.2 Contributions

The key contributions of this chapter are:

1. We introduce XBRL tagging, a new financial NLP task for a real-world need, and we release FiNER-139, the first XBRL tagging dataset, containing 1M sentences with XBRL labels.[2]

2. We provide extensive benchmarking experiments using BILSTMs and BERT with generic or in-domain pre-training, which establish strong baseline results for future work on FiNER-139. We also compare with Claude Sonnet 4 by Anthropic, a state-of-the-art Large Language Model, and show that LLMs struggle significantly in such a task with many labels and very specific financial terminology.

3. We demonstrate that substituting numeric tokens with pseudo-tokens encoding their morphological patterns and approximate magnitudes leads to substantial performance improvements in BERT-based models and LLMs for this task. This tokenization strategy proves critical for enabling language models to effectively process and leverage numerical information.

4. We release a new family of BERT models (SEC-BERT, SEC-BERT-NUM, SEC-BERT-SHAPE) pre-trained on 200k financial filings, incorporating our proposed tokenization method, obtaining the best results on FiNER-139.[3][4][5]

[2] https://huggingface.co/datasets/nlpaueb/finer-139
[3] https://huggingface.co/nlpaueb/sec-bert-base
[4] https://huggingface.co/nlpaueb/sec-bert-num
[5] https://huggingface.co/nlpaueb/sec-bert-shape

5. Building on all the previous items, we also compile a system describing the above in a granted US patent (Loukas et al., 2021e), as well as corresponding patent application in the European Patent Office and the World Intellectual Property Organization.[6]

3.3 Related Work

3.3.1 Entity Extraction

XBRL tagging differs from NER and other previous entity extraction tasks (Table 3.1), like contract element extraction (Chalkidis et al., 2019). Crucially, in XBRL tagging there is a much larger set of entity types (6k in full XBRL, 139 in FiNER-139), most tagged tokens are numeric (~91%), and the correct tag highly depends on context. In most NER datasets, numeric expressions are classified in generic entity types like 'amount' or 'date' (Bikel et al., 1999); this can often be achieved with regular expressions that look for common formats of numeric expressions, and the latter are often among the easiest entity types in NER datasets. By contrast, although it is easy to figure out that the first three highlighted expressions of Figure 3.1 are amounts, assigning them the correct XBRL tags requires carefully considering their context. Contract element extraction (Chalkidis et al., 2019) also requires considering the context of dates, amounts etc. to distinguish, for example, start dates from end dates, total amounts from other mentioned amounts, but the number of entity types in FiNER-139 is an order of magnitude larger (Table 3.1) and the full tag set of XBRL is even larger (6k).

[6]More details can be found at the original document.

Dataset	Domain	Entity Types
CONLL-2003	Generic	4
ONTONOTES-V5	Generic	18
ACE-2005	Generic	7
GENIA	Biomedical	36
Chalkidis et al. (2019)	Legal	14
Francis et al. (2019)	Financial	9
Shah et al. (2024)	Financial	3
FiNER-139 (ours)	Financial	**139**

Table 3.1: Examples of previous entity extraction datasets. Information about the first four from Tjong Kim Sang and De Meulder (2003); Pradhan et al. (2012); Doddington et al. (2004); Kim et al. (2003).

3.3.2 Named Entity Recognition in Finance

Applications of NER in finance have previously employed at most 9 (generic) class labels. Salinas Alvarado et al. (2015) investigated NER in finance to recognize organizations, persons, locations, and miscellaneous entities on 8 manually annotated SEC financial agreements using CRFs. Francis et al. (2019) experimented with transfer learning by unfreezing different layers of a BILSTM with a CRF layer, pre-trained on invoices, to extract 9 entity types with distinct morphological patterns (e.g., IBAN, company name, date, total amount). Also, Hampton et al. (2015, 2016) applied a Maximum Entropy classifier, CRFs, and handcrafted rules to London Stock Exchange filings to detect 9 generic entity types (e.g., person, organization, location, money, date, percentages). Kumar et al. (2016) extended the work of Finkel et al. (2005) and built a financial entity recognizer of dates, numeric values, economic terms in SEC and non-SEC documents, using numerous handcrafted text features. More recently, Lu and Huo (2025) presented a systematic evaluation of state-of-the-art LLMs and prompting techniques in financial documents in the dataset of Shah et al. (2024), which has 201 annotated financial articles with the standard organization/person/location entities. By contrast, FiNER-139 uses a specialized set of 139 highly technical economic tags derived from the real-world need of XBRL tagging, and we employ no handcrafted features.

3.3.3 Numerical Reasoning

Research in neural numerical reasoning focuses on how to represent numbers to solve numeracy tasks, e.g., compare numbers, understand mathematical operations mentioned in a text etc. Zhang et al. (2020) released NUMBERT, a Transformer-based model that handles numerical reasoning tasks by representing numbers by their scientific notation and applying subword tokenization. On the other hand, GENBERT (Geva et al., 2020) uses the decimal notation and digit-by-digit tokenization of numbers. Both models attempt to deal with the problem that word-level tokenization often turns numeric tokens to OOVs (Thawani et al., 2021). This is important, because numerical reasoning requires modeling the exact value of each numeric token. In FINER-139, the correct XBRL tags of numeric tokens depend much more on their contexts and token shapes than on their exact numeric values (Fig. 3.1). Hence, these methods are not directly relevant. GENBERT's digit-by-digit tokenization would also lead to excessive fragmentation, which we experimentally find to harm performance.

3.4 Task and Dataset

Historically, filings from business organizations were simply rendered in plain text (.txt) files. Thus, analysts and researchers needed to manually identify, copy, and paste each amount of interest (e.g., from filings to spreadsheets). With XBRL-tagged filings, identifying and extracting amounts of interest (e.g., to spreadsheets or databases) can be automated. More generally, XBRL facilitates the machine processing of financial documents. Hence, XBRL-tagged financial reports are required in several countries, as already noted (Section 3.1). However, manually tagging reports with XBRL tags is tedious and resource-intensive. Therefore, we release the FINER-139 dataset to foster research towards automating XBRL tagging.

We constructed FINER-139 from approximately 10k annual and quarterly English reports (filings) of publicly traded companies downloaded from SEC's EDGAR system.[7] The downloaded reports span a 5-year period, from 2016 to 2020. They are annotated with XBRL tags by professional auditors and describe the performance and projections of the companies. We used regular expressions to extract the text notes from the *Financial Statements Item* of each filing, which is the primary source of XBRL tags in annual and quarterly reports.

XBRL taxonomies have many different attributes, making XBRL tagging challenging even for humans (Baldwin et al., 2006; Hoitash and Hoitash, 2018). Furthermore, each juris-

[7] https://www.sec.gov/edgar/

diction has its own XBRL taxonomy. Since we work with US documents, our labels come from the U.S. Generally Accepted Accounting Principles (GAAP).[8] Since this is the first effort towards automatic XBRL tagging, we chose to work with the most essential and informative attribute, the *tag names*, which populate our label set. Each tagged fact in XBRL includes not only the tag name (i.e., concept), but also attributes such as the reporting context (entity and period), the measurement unit (e.g., USD or shares), the reported value, the number of decimals, and, if applicable, dimensional breakdowns (e.g., by segment or geography). These attributes are embedded in Inline XBRL (iXBRL) using HTML-like tags, allowing the fact to be both human-readable and machine-processable. This XML-like technical format involves verbose constructs like `contextRef="c1"` (reporting entity and period), `tagname="us-gaap:EntityName"` (tags, which we use in our study), `unitRef="u1"` (unit), `decimals="6"` (numeric precision), and optionally `xml:lang="en-US"`, making even simple facts structurally complex to parse. Also, since XBRL tags change periodically, we selected the 139 (out of 6,008) most frequent XBRL tags with at least 1,000 appearances in FiNER-139. The distribution of these tags seems to follow a power law (Figure 3.2), hence most of the 6k XBRL tags that we did not consider are very rare. We used the IOB2 annotation scheme to distinguish tokens at the beginning, inside, or outside of tagged expressions, which leads to 279 possible token labels.

Splitting the text notes resulted in 1.8M sentences, the majority of which (~90%) contained no tags.[9] The sentences are also HTML-stripped, normalized, and lower-cased. To avoid conflating trivial and more difficult cases, we apply heuristic rules to discard sentences that can be easily flagged as almost certainly requiring no tagging; in a real-life setting, the heuristics, possibly further improved, would discard sentences that do not need to be processed by the tagger. The heuristic rules were created by inspecting the training subset and include regular expressions that look for amounts and other expressions that are typically annotated. In total, these heuristics removed about 40% of the 1.8M sentences, while affecting only 1% of the sentences that were actually tagged. We split chronologically the remaining sentences into training, development, and test sets with an 80/10/10 ratio (Table 3.2).

[8] `www.xbrl.us/xbrl-taxonomy/2020-us-gaap/`

[9] We use NLTK (Bird et al., 2009) for sentence splitting.

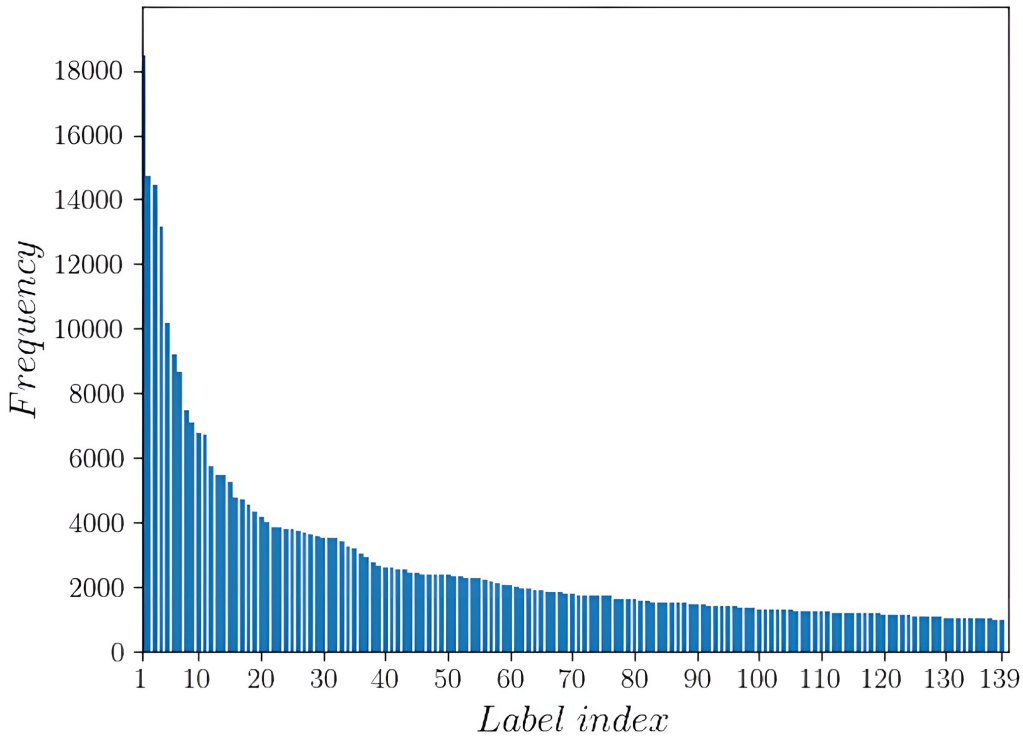


Figure 3.2: Frequency distribution of the 139 XBRL tags used in this work over the entire FiNER-139 dataset. Label indices shown instead of tag names to save space.

Subset	Sentences (S)	Avg. Tokens/S	Avg. Tags/S
Train	900,384	44.7 ± 33.9	1.8 ± 1.2
Dev	112,494	45.4 ± 35.9	1.7 ± 1.2
Test	108,378	46.5 ± 38.9	1.7 ± 1.1

Table 3.2: FiNER-139 statistics, using SPACY's tokenizer and the 139 tags of this work (± standard deviation).

3.5 Baseline Models

We start with using several baseline models for the XBRL tagging task:

- spaCy (Honnibal et al., 2020) is an open-source and industry-ready NLP library.[10] It includes a NER component that uses word-level Bloom embeddings (Serrà and Karatzoglou, 2017) and residual Convolutional Neural Networks (CNNS) (He et al., 2016). We trained spaCy's NER tool from scratch on FiNER-139.

[10]We used SPACY v.2.3; see `https://spacy.io/`.

- The second baseline employs a stacked bidirectional Long-Short Term Memory (LSTM) network (Graves et al., 2013; Lample et al., 2016a) with residual connections. Each token t_i of a sentence S is mapped to an embedding and passed through the BILSTM stack to extract the corresponding contextualized embedding. A shared multinomial logistic regression (LR) layer operates on top of each contextualized embedding to predict the correct label. We use the EDGAR-W2V embeddings we created in Section 2.5.2 (Loukas et al., 2021b).

- We fine-tune a BERT-BASE model (Devlin et al., 2019a) to extract contextualized embeddings of subwords. Again, a multinomial LR layer operates on top of the contextualized embeddings to predict the correct label of the corresponding subword.

- Lastly, we replace the LR layer of the BILSTM and BERT models with a Conditional Random Field (CRF) layer (Lafferty et al., 2001). This layer has been shown to be beneficial in several token labeling tasks (Huang et al., 2015; Lample et al., 2016b; Ma and Hovy, 2016; Lample et al., 2016a; Chalkidis et al., 2020c).[11] A Conditional Random Field (CRF) layer models the conditional probability distribution over label sequences given the input sequence, taking into account dependencies between adjacent labels. Unlike the multinomial logistic regression layer which makes independent predictions for each token, a linear-chain CRF considers the entire label sequence when making predictions.

  For a sentence $S = \{t_1, t_2, \ldots, t_n\}$ with the corresponding contextualized embeddings $\{h_1, h_2, \ldots, h_n\}$ from the BILSTM or BERT encoder, we define two types of potentials: the emission potential $\psi_i(y_i, h_i)$, which measures how likely label y_i is given the contextualized embedding h_i, and the transition potential $\psi_{i,i+1}(y_i, y_{i+1})$, which captures dependencies between consecutive labels. The probability of a label sequence $y = \{y_1, y_2, \ldots, y_n\}$ is then defined as:

$$P(y \mid S) = \frac{\exp\Big(\sum_{i=1}^{n} \psi_i(y_i, h_i) + \sum_{i=1}^{n-1} \psi_{i,i+1}(y_i, y_{i+1})\Big)}{Z(S)} \tag{3.1}$$

  The denominator $Z(S)$ is the partition function, which sums over all possible label sequences of length n:

[11] We use a linear-chain CRF layer with log-likelihood optimization and Viterbi decoding.

$$Z(S) = \sum_{y'} \exp\left(\sum_{i=1}^{n} \psi_i(y'_i, h_i) + \sum_{i=1}^{n-1} \psi_{i,i+1}(y'_i, y'_{i+1}) \right) \tag{3.2}$$

Since enumerating all sequences is computationally infeasible, $Z(S)$ is computed efficiently using dynamic programming. Specifically, the forward-backward algorithm is used (Baum et al., 1970). In practice, the forward recursion suffices to obtain the partition function, while the full forward–backward algorithm is used during training to compute label marginals required for gradient updates. In contrast, at inference time, when we only need the single best label sequence, we use the Viterbi algorithm (Viterbi, 1967) to obtain $\hat{y} = \arg\max_y P(y \mid S)$.

During training, the model minimizes the negative log-likelihood loss:

$$\mathcal{L} = -\sum_{k=1}^{K} \log P(y^{(k)}|S^{(k)}) \tag{3.3}$$

where K is the number of training examples, and $P(y^{(k)}|S^{(k)})$ is the probability assigned to the ground-truth labels of the k-th training sentence.

3.6 Baseline Results

We report micro-F_1 (μ-F_1) and macro-F_1 (m-F_1) at the entity level, i.e., if a gold tag annotates a multi-word span, a model gets credit only if it tags the exact same span. This allows comparing more easily methods that label words vs. subwords. μ-F_1 gives equal weight to each individual entity occurrence, favoring frequent tags, while m-F_1 computes the average F_1 score across tags, treating all tags equally regardless of their frequency. Also, to extract better and generalizable insights, we run all of the experiments for 3 runs using random seeds and report average scores.

Table 3.3 reveals that SPACY performs poorly, possibly due to the differences from typical token labeling tasks, i.e., the large amount of entity types, the abundance of numeric tokens, and the fact that in FINER-139 the tagging decisions depend mostly on context. Interestingly enough, BILSTM with word embeddings (underlined in Table 3.3) performs slightly better than BERT. However, when a CRF layer is added, BERT achieves the best results, while the performance of BILSTM (with word embeddings) deteriorates significantly, contradicting previous studies (Ma and Hovy, 2016; Lin et al., 2017).

Baseline methods	μ-F_1	m-F_1
SPACY v2.3 (words)	48.6 ± 0.4	37.6 ± 0.2
BILSTM (words)	<u>77.3</u> ± 0.6	<u>73.8</u> ± 1.8
BILSTM (subwords)	71.3 ± 0.2	68.6 ± 0.2
BERT (subwords)	75.1 ± 1.1	72.6 ± 1.4
BILSTM (words) + CRF	69.4 ± 1.2	67.3 ± 1.6
BILSTM (subwords) + CRF	76.2 ± 0.2	73.4 ± 0.3
BERT (subwords) + CRF	**78.0** ± 0.5	**75.2** ± 0.6

Table 3.3: Entity-level μ-F_1 and m-F_1 (%, avg. of 3 runs with different random seeds, ± std. dev.) on test data. Surprisingly, BILSTM with word embeddings (underlined) performs slightly better than BERT. Additional detailed results can be found in Appendix B.

We hypothesize that the inconsistent effect of CRFs is due to tokenization differences. When using BERT's subword tokenizer, there are more decisions that need to be all correct for a tagged span to be correct (one decision per subword) than when using word tokenization (one decision per word). Thus, it becomes more difficult for subword models to avoid nonsensical sequences of token labels, e.g., labeling two consecutive subwords as beginning and inside of different entity types, especially given the large set of 279 labels (Table 3.1).

Let's consider this example: *"The Company's net revenue increased to 199.5 million dollars for the fiscal year ended December 31, 2023."*. Here, the financial amount "199.5" is tokenized by BERT into subwords ["199", ".", "5"], which without proper sequence modeling can lead to invalid tag sequences such as `[B-Revenues]` followed by `[I- Revenue From Contract With Customer Excluding Assessed Tax]`, `[I- Revenue From Contract With Customer Excluding Assessed Tax]`, a clear tagging violation where the "inside" tags belong to a completely different financial concept than the "beginning" tag. Such nonsensical sequences are impossible with word-level tokenization, where "199.5" is a single token and it would receive a single tag, representing one unified financial amount. Thus, the CRF layer on top of subword models helps reduce these nonsensical sequences by enforcing transition constraints that penalize invalid combinations across different entity types, ensuring that subword fragments of the same semantic unit (like a monetary amount) receive consistent labeling.

Conversely, when utilizing words as tokens, fewer opportunities exist for nonsensical label

sequences, because there are fewer tokens. For instance, the average number of subwords and words per gold span is 2.53 and 1.04, respectively. Hence, it is easier for the BILSTM to avoid predicting nonsensical sequences of labels and the CRF layer on top of the BILSTM (with word embeddings) has less room to contribute and mainly introduces noise (e.g., it often assigns low probabilities to acceptable, but less frequent label sequences). With the CRF layer, the model tries to maximize both the confidence of the BILSTM for the predicted label of each word and the probability that the predicted sequence of labels is frequent. When the BILSTM on its own rarely predicts nonsensical sequences of labels, adding the CRF layer rewards commonly seen sequences of labels, even if they are not the correct labels, without reducing the already rare nonsensical sequences of labels.

To further support our hypothesis, we repeated the BILSTM experiments, but with subword (instead of word) embeddings, trained on the same vocabulary as BERT. Without the CRF, the subword BILSTM performs much worse than the word BILSTM (6 p.p drop in μ-F_1), because of the many more decisions and opportunities to predict nonsensical label sequences. The CRF layer substantially improves the performance of the subword BILSTM (4.9 p.p. increase in μ-F_1), as expected, though the word BILSTM (without CRF) is still better, because of the fewer opportunities for nonsensical predictions.

A noted drawback of CRFs is that they significantly slow down the models both during training and inference (Teichmann and Cipolla, 2018), especially when using large label sets (Goldman and Goldberger, 2020), as in our case, since they scale quadratically with the label set size. Hence, although BERT with CRF was the best model in Table 3.3, in the following sections we focus on how to improve BERT's performance further without employing CRFs.

3.7 Numeric Fragmentation in BERT

Motivated by BiLSTM's superior performance over BERT, we dig inside and analyze the sentences in FiNER-139. There, we see that the majority (91.2%) of the gold tagged spans are numeric expressions, which cannot all be included in BERT's finite vocabulary. For instance, the token '9,323.0' is split into five subword units, ['9', '##,', '##323', '##.', '##0'], while the token '12.78' is split into ['12', '##.', '##78']. The excessive fragmentation of numeric expressions, when using subword tokenization, harms the performance of the subword-based models (Table 3.3), because it increases the probability of producing nonsensical sequences of labels, as already discussed. This means the model must correctly predict labels for multiple subword pieces to identify a single numeric entity. Moreover, different numeric formats fragment differently, creating inconsistent patterns the model must learn. We, therefore, propose two simple and effective solutions to avoid the over-fragmentation of numbers.

1. BERT + [NUM]: We detect all numbers using regular expressions and replace each one with a single [NUM] pseudo-token, which cannot be split. This single pseudo-token is added to the BERT vocabulary, and its representation is learned during fine-tuning. This allows handling all numeric expressions in the same manner, disallowing their fragmentation.

2. BERT + [SHAPE]: Our second approach replaces numbers with pseudo-tokens that cannot be split and represent the number's shape (or else called magnitude). For instance, '53.2' becomes '[XX.X]', and '40,200.5' becomes '[XX,XXX.X]'. We use 214 special tokens that cover all the number shapes we find in the training set. Again, the representations of the pseudo-tokens are fine-tuned, and numeric expressions (of known shapes) are no longer fragmented. The shape pseudo-tokens also capture information about each number's magnitude. The intuition behind that is that numeric tokens of similar magnitudes may require similar XBRL tags. Figure 3.3 illustrates the use of [NUM] and [SHAPE] pseudo-tokens.

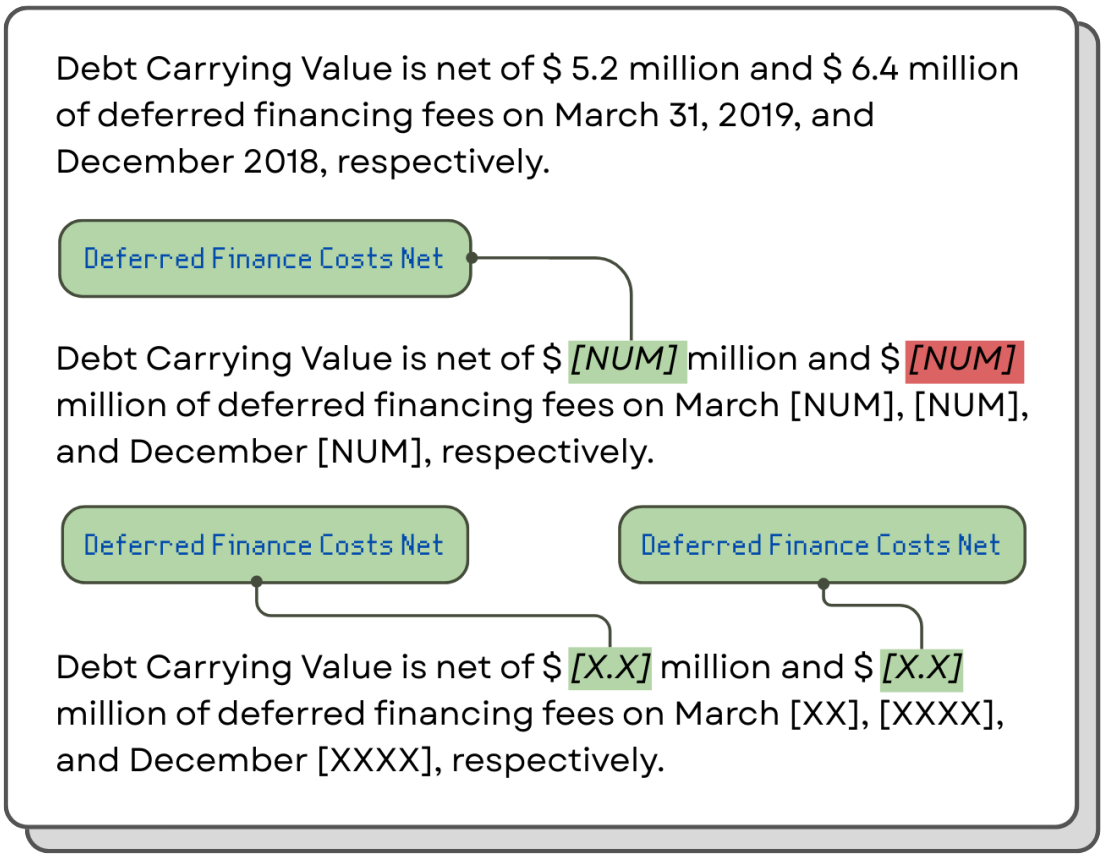


Figure 3.3: XBRL tag predictions of BERT (top), BERT + [NUM] (middle), BERT + [SHAPE] (bottom) for the same sentence. BERT tags incorrectly the amounts in red. BERT + [NUM] and BERT + [SHAPE] tag them more successfully (green indicates correct tags).

3.8 In-domain Pre-training

Driven by the findings that pre-training language models on specialized domains is beneficial for downstream tasks (Alsentzer et al., 2019; Beltagy et al., 2019; Yang et al., 2020; Chalkidis et al., 2020c), we explore this direction in our task which is derived from the financial domain.

- **FIN-BERT:** We first fine-tune FIN-BERT (Yang et al., 2020), which is pre-trained on a financial corpus from SEC documents, earnings call transcripts, and analyst reports.[12] The 30k subwords vocabulary of FIN-BERT is built from scratch from its pre-training corpus. Again, we utilize FIN-BERT with and without our numeric pseudo-tokens, whose representations are learned during fine-tuning.

[12]We use the FINBERT-FINVOCAB-UNCASED version from `https://github.com/yya518/FinBERT`.

- **SEC-BERT:** We also release our own family of BERT models. Following the original setup of Devlin et al. (2019a), we pre-trained BERT from scratch on EDGAR-CORPUS, a collection of financial documents we released previously in Section 2.5 (Loukas et al., 2021b). The resulting model, which we SEC-BERT, has a newly created vocabulary of 30k subwords. Additionally, to further examine the impact of the proposed [NUM] and [SHAPE] special tokens, we also pre-trained two additional BERT variants, SEC-BERT-NUM and SEC-BERT-SHAPE, on the same corpus, having replaced all numbers by [NUM] or [SHAPE] pseudo-tokens, respectively. In this case, the representations of the pseudo-tokens are learned both during pre-training and they are also updated during fine-tuning.

3.9 Improved BERT Results

Table 3.4 reports micro-averaged precision, recall, and F_1 on development and test data. As with Table 3.3, a LR layer is used on top of each embedding to predict the correct label, unless specified otherwise.

	DEVELOPMENT			TEST		
	μ-P	μ-R	μ-F_1	μ-P	μ-R	μ-F_1
BERT	74.9 ± 1.5	82.0 ± 1.3	78.2 ± 1.4	71.5 ± 1.1	79.6 ± 1.4	75.1 ± 1.1
BERT + CRF	78.3 ± 0.8	83.6 ± 0.4	80.9 ± 0.3	75.0 ± 0.9	81.2 ± 0.2	78.0 ± 0.5
BERT + [NUM]	79.4 ± 0.8	83.0 ± 0.9	81.2 ± 0.9	76.0 ± 0.6	80.7 ± 0.8	78.3 ± 0.7
BERT + [SHAPE]	82.1 ± 0.6	82.6 ± 0.4	82.3 ± 0.2	78.7 ± 0.5	80.1 ± 0.2	79.4 ± 0.2
FIN-BERT	73.9 ± 1.3	81.4 ± 0.7	77.5 ± 1.0	70.2 ± 1.2	78.7 ± 0.7	74.0 ± 1.1
FIN-BERT + [NUM]	81.1 ± 0.1	82.5 ± 1.2	81.8 ± 0.1	77.9 ± 0.1	79.9 ± 0.7	78.8 ± 0.3
FIN-BERT + [SHAPE]	82.3 ± 1.7	84.0 ± 1.2	83.2 ± 1.4	79.0 ± 1.6	81.2 ± 1.1	80.1 ± 1.4
SEC-BERT (ours)	75.2 ± 0.4	82.7 ± 0.5	78.8 ± 0.1	71.6 ± 0.4	80.3 ± 0.5	75.7 ± 0.1
SEC-BERT-NUM (ours)	82.5 ± 2.1	84.4 ± 1.2	83.7 ± 1.7	79.0 ± 1.9	82.0 ± 0.9	80.4 ± 1.4
SEC-BERT-SHAPE (ours)	**84.8** ± 0.2	**85.8** ± 0.2	**85.3** ± 0.0	**81.0** ± 0.2	**83.2** ± 0.1	**82.1** ± 0.1

Table 3.4: Entity-level micro-averaged P, R, F_1 ± std. dev. (3 runs) on the dev. and test data for BERT-based models. Underlining denotes the best metrics within each group, while bold denotes the best metric across all different groups.

Focusing on the second zone of the table, we observe that the [NUM] pseudo-token im-

proves BERT's results, as expected, since it does not allow numeric expressions to be fragmented. The results of BERT + [NUM] are now comparable to those of BERT + CRF. Performance improves further when utilizing the shape pseudo-tokens (BERT + [SHAPE]), yielding 79.4 μ-F_1 and showing that information about each number's magnitude is valuable in XBRL tagging.

Interestingly, FIN-BERT (3rd zone) performs worse than BERT despite its pre-training on financial data. Similarly to BERT, this can be attributed to the fragmentation of numbers (2.5 subwords per gold tag span). Again, the proposed pseudo-tokens ([NUM], [SHAPE]) alleviate this problem and allow FIN-BERT to leverage its in-domain pre-training in order to finally surpass the corresponding BERT variants, achieving an 80.1 μ-F_1 test score.

Our new model, SEC-BERT (last zone), pre-trained on SEC reports, performs better than the existing BERT and FIN-BERT models when no numeric pseudo-tokens are used. However, SEC-BERT is still worse than BERT with numeric pseudo-tokens (75.7 vs. 78.3 and 79.4 test μ-F_1), suffering from number fragmentation (2.4 subwords per gold tag span). SEC-BERT (without pseudo-tokens) also performs worse than the BILSTM with word embeddings (75.7 vs. 77.3 μ-F_1, cf. Table 3.3). However, when the proposed pseudo-tokens are used, SEC-BERT-NUM and SEC-BERT-SHAPE achieve the best overall performance, boosting the test μ-F_1 to 80.4 and 82.1, respectively. This indicates that learning to handle numeric expressions during model pre-training is a better strategy than trying to acquire this knowledge only during fine-tuning.

3.10 Experimental Setup

For SPACY, we followed the recommended practices.[13] All other methods were implemented in TENSORFLOW.[14] Concerning BERT models, we used the implementation of HUGGINGFACE (Wolf et al., 2020). We also use Adam (Kingma and Ba, 2015), Glorot initialization (Glorot and Bengio, 2010), and the categorical cross-entropy loss.

Hyper-parameters were tuned on development data with Bayesian Optimization (Snoek et al., 2012) monitoring the development loss for 15 trials.[15] For the BILSTM encoders, we searched for $\{1, 2, 3\}$ hidden layers, $\{128, 200, 256\}$ hidden units, $\{1e\text{-}3, 2e\text{-}3, 3e\text{-}3, 4e\text{-}3, 5e\text{-}3\}$ learning rate, and $\{0.1, 0.2, 0.3\}$ dropout. We trained for 30 epochs using early stopping with

[13] `https://spacy.io/usage/v2-3`.

[14] `https://www.tensorflow.org/`

[15] We used KERAS TUNER (`https://keras-team.github.io/keras-tuner/documentation/tuners/`)

	Params	*L*	*U*	P_{drop}	*LR*
BILSTM (words)	21M	2	128	0.1	1e-3
BILSTM (subwords)	8M	1	256	0.2	1e-3
BILSTM (words) + CRF	21M	2	128	0.1	1e-3
BILSTM (subwords) + CRF	8M	1	256	0.2	1e-3
BILSTM-NUM (subwords)	1M	1	256	0.2	1e-3
BILSTM-SHAPE (subwords)	0.8M	2	128	0.1	1e-3
BERT	110M	-	-	-	1e-5
BERT + [NUM]	110M	-	-	-	1e-5
BERT + [SHAPE]	110M	-	-	-	1e-5
BERT + CRF	110M	-	-	-	1e-5
FIN-BERT	110M	-	-	-	2e-5
FIN-BERT + [NUM]	110M	-	-	-	2e-5
FIN-BERT + [SHAPE]	110M	-	-	-	2e-5
SEC-BERT	110M	-	-	-	1e-5
SEC-BERT-NUM	110M	-	-	-	1e-5
SEC-BERT-SHAPE	110M	-	-	-	1e-5

Table 3.5: Number of total parameters (*Params*) and the best hyper-parameter values for each method, i.e., number of recurrent layers (*L*), number of recurrent units (*U*), dropout probability P_{drop}, learning rate (*LR*).

patience 4. For BERT, we used grid-search to select the optimal learning rate from $\{1e\text{-}5, 2e\text{-}5, 3e\text{-}5, 4e\text{-}5, 5e\text{-}5\}$, fine-tuning for 10 epochs, using early stopping with patience 2. All final hyper-parameters are shown in Table 3.5. Training was performed mainly on a DGX station with 4 NVIDIA V100 GPUs and an Intel Xeon CPU E5-2698 v4 @ 2.20GHz.

3.11 Additional Experiments

3.11.1 Subword pooling

A different approach to avoid word fragmentation is to employ subword pooling for each individual word. Ács et al. (2021) found that for NER tasks, it is better to use the *first* subword only, i.e., predict the label of an entire word from the contextualized embedding of its first subword only; they compared to several other methods, such as using only the *last* subword of

each word, or combining the contextualized embeddings of all subwords with a self-attention mechanism. Given this finding, we conducted an ablation study and compare (i) our best model (SEC-BERT) with *first* subword pooling (denoted SEC-BERT-FIRST) to (ii) SEC-BERT with our special tokens (SEC-BERT-NUM, SEC-BERT-SHAPE), which avoid segmenting numeric tokens.

Table 3.6 demonstrates that, in XBRL tagging, using the proposed special tokens is comparable (SEC-BERT-NUM) or better (SEC-BERT-SHAPE) than performing *first* pooling (SEC-BERT-FIRST).

	μ-F_1	m-F_1
SEC-BERT	78.8 ± 0.1	72.6 ± 0.4
SEC-BERT-FIRST	79.9 ± 1.2	77.1 ± 1.7
SEC-BERT-NUM	80.4 ± 1.4	78.9 ± 1.3
SEC-BERT-SHAPE	**82.1** ± 0.1	**80.1** ± 0.2

Table 3.6: Entity-level μ-F_1 and m-F_1 (%, avg. of 3 runs with different random seeds, ± std. dev.) on test data using different ways to alleviate fragmentation.

3.11.2 Subword BiLSTM with [NUM] and [SHAPE]

To further investigate the effectiveness of our pseudo-tokens, which encode morphological patterns and approximate magnitudes, we incorporated them in the BILSTM operating on subword embeddings (3rd model of Table 3.3). Again, we replace each number by a single [NUM] pseudo-token or one of 214 [SHAPE] pseudo-tokens, for the two approaches, respectively. These replacements also happen when pre-training WORD2VEC subword embeddings; hence, an embedding is obtained for each pseudo-token.

Table 3.7 indicates that BILSTM-NUM outperforms the BILSTM subword model. BILSTM-SHAPE further improves performance and is the best BILSTM subword model overall, surpassing the subword BILSTM with CRF, which was the best subword BILSTM model in Table 3.3. These results further support our hypothesis that the [NUM] and [SHAPE] pseudo-tokens help subword models successfully generalize over numeric expressions, with [SHAPE] being the best of the two approaches, while also avoiding the over-fragmentation of numbers.

	μ-F_1	m-F_1
BILSTM (subwords)	71.3 ± 0.2	68.6 ± 0.2
BILSTM (subwords) + CRF	76.2 ± 0.2	73.4 ± 0.3
BILSTM-NUM (subwords)	75.6 ± 0.3	72.7 ± 0.4
BILSTM-SHAPE (subwords)	**76.8** ± 0.2	**74.1** ± 0.3

Table 3.7: Entity-level μ-F_1 and m-F_1 (%, avg. of 3 runs with different random seeds, ± std. dev.) on test data for BILSTM models with [NUM] and [SHAPE] tokens. Additional detailed results can be found in Appendix B.

3.11.3 A Business Use Case

Since XBRL tagging is derived from a real-world need, it is crucial to analyze the model's performance in a business use case. After consulting with experts of the financial domain, we concluded that one practical use case would be to use an XBRL tagger as a recommendation engine that would propose the k most probable XBRL tags for a specific token selected by the user. The idea is that an expert (e.g., accountant, auditor) knows beforehand the token(s) that should be annotated and the tagger would assist by helping identify the appropriate tags more quickly. Instead of having to select from several hundreds of XBRL tags, the expert would only have to inspect a short list of k proposed tags.

We evaluate our best model, SEC-BERT-SHAPE, in this scenario using Hits@k. We use the model to return the k most probable XBRL tags for each token that needs to be annotated, now assuming that the tokens to be annotated are known. If the correct tag is among the top k, we increase the number of hits by one. Finally, we divide by the number of tokens to be annotated. Figure 3.4 shows the results for different values of k. The curve is steep for $k = 1$ to 5 and saturates as k approaches 10, where Hits@k is nearly perfect (99.4%). In practice, this means that a user would have to inspect 10 recommended XBRL tags instead of hundreds for each token to be annotated; and in most cases, the correct tag would be among the top 5 recommended ones.

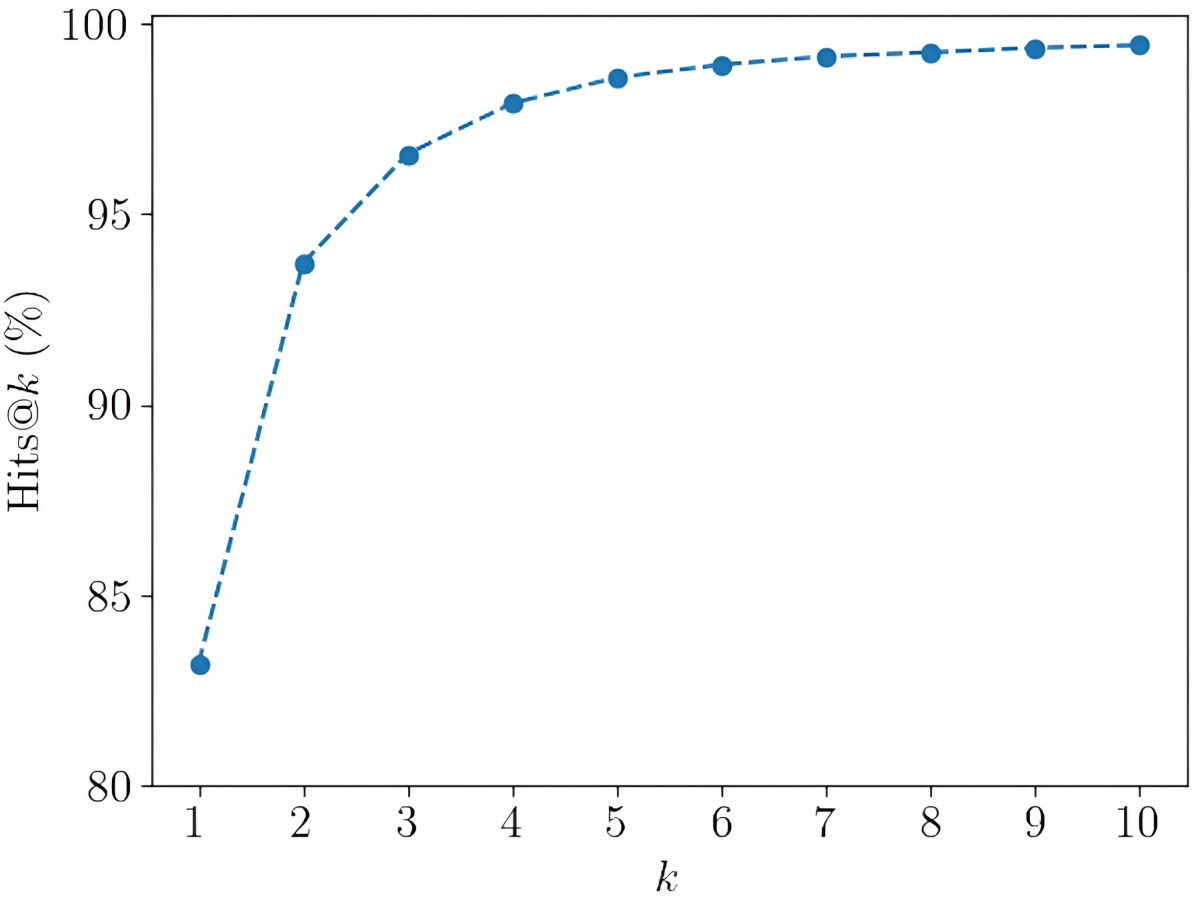


Figure 3.4: Hits@k results (%, average of 3 runs with different random seeds) on test data, for different k values. Standard deviations were very small and are omitted.

3.12 Error Analysis

To unveil the peculiarities of FiNER-139, extract new insights about it, and discover any limitations, we also performed an exploratory data and error analysis. Specifically, we manually inspected the errors of SEC-BERT-SHAPE (the best performing model) in under-performing classes (where $F_1 < 50\%$) and identified three main sources of errors.

The first category of errors is related to **specialized terminology**. In this category, the model is able to understand the general financial semantics, but does not fully comprehend highly technical details. For example, *Operating Lease Expense* amounts are sometimes misclassified as *Lease And Rental Expense*, i.e., the model manages to predict that these amounts are about expenses in general, but fails to identify the specific details that distinguish operating lease expenses from lease and rental expenses. Similarly, *Payments to Acquire Businesses (Net of Cash Acquired)* amounts are mostly misclassified as *Payments to Acquire Businesses*

(Gross). In this case, the model understands the notion of business acquisition, but fails to differentiate between net and gross payments.

Another interesting error type involves the misclassification of **financial dates**. For example, tokens of the class *Debt Instrument Maturity Date* are mostly missclassified as not belonging to any entity at all (*'O'* tag). Given the previous type of errors, one would expect the model to missclassify these tokens as a different type of financial date, but this is not the case here. We suspect that errors of this type may be due to annotation inconsistencies by the financial experts.

Additionally, although the gold XBRL tags of FiNER-139 originate from professional auditors, as required by the Securities & Exchange Commission (SEC) legislation, there are still some **annotation discrepancies**. We provide an illustrative example in Figure 3.5. We believe that such inconsistencies are inevitable to occur and they are a part of the real-world nature of the problem.

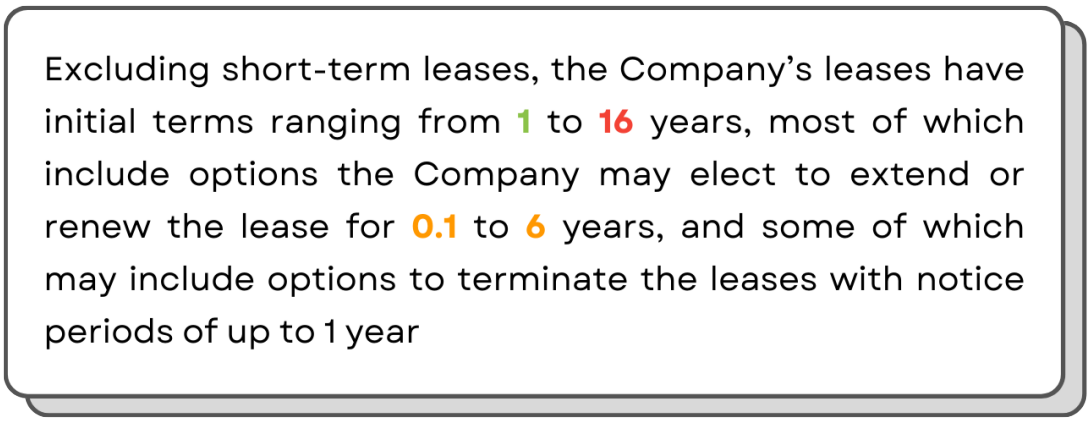


Figure 3.5: A manually inspected sentence from FiNER-139 showing some inconsistencies in the gold XBRL tags of the auditors. The green '1' is correctly annotated with the XBRL tag *Lessee Operating Lease Term Of Contract*. The red '16' should have also been annotated with the same tag, but is not, possibly because the annotator thought the (same) tag was obvious. The orange numbers '0.1' and '6' lack XBRL annotations; they should have both been annotated as *Lessee Operating Lease Renewal Term*. We can only speculate that the auditor might not have been aware that there is an XBRL tag for lease renewal terms, in which case the recommendation engine of Section 3.11.3 might have helped.

We anticipate that this analysis will motivate subsequent research on classification tasks

with numeric tokens, like XBRL tagging. In more detail, for this task, we believe that the specialized terminology and financial date errors may be alleviated by adopting hierarchical classifiers (Chalkidis et al., 2020b; Manginas et al., 2020), which would first detect entities in coarse classes (e.g., expenses, dates) and would then try to classify the identified entities into finer classes (e.g., lease vs. rent expenses, instrument maturity dates vs. other types of dates). It would also be interesting to train classifiers towards detecting wrong (or missing) gold annotations, in order to help in quality assurance checks of XBRL-tagged documents.

3.13 Impact in other scientific studies

Our published work on XBRL tagging with the FiNER-139 dataset (Loukas et al., 2022) motivated in the following years researchers from Goldman Sachs to extend our study. Building directly on our work, Sharma et al. (2023) introduced the Financial Numeric Extreme Labelling ("FNXL") dataset. While FiNER-139 (ours) focused on the 139 most frequent XBRL tags in order to introduce the task, FNXL expands the label set.

The FNXL dataset was created from SEC 10-K filings only[16] (2019-2021). It contains approximately 79k sentences, compared to FiNER-139's 1M sentences, but contain more annotated numerals, specifically 143,000 ones. Their label set consists of 2,794 distinct labels, compared to FiNER-139's 139 entity labels. Their label set follows a long-tail distribution too, similar to FiNER-139, just on a much larger scale. The dataset was split based on companies to prevent data leakage between training, validation, and test sets.

In their work, Sharma et al. (2023) proposed a two-stage pipeline suitable for extreme classification: (1) a binary BERT tagger identifies relevant numerals in a sentence, and (2) an AttentionXML model (You et al., 2018, 2019) assigns the XBRL label to the identified numeral. The binary tagger is essentially a BERT-based sequence tagger trained on sentences (from the train split) with numeric tokens that need to be labeled with an XBRL label or not. This binary classification task simplifies the entire process by narrowing down the numerals that require further processing for label assignment.

The results of Sharma et al. (2023), which are summarized in Table 3.8), showed that the AttentionXML pipeline achieved slightly better scores when using our proposed pseudo-token ([SHAPE]). Incorporating the [NUM] token did not help their standard AttentionXML baseline. Additionally, their bucket analysis confirmed AttentionXML's strength on labels

[16] As a reminder, FiNER-139 consisted of multiple 10-K and 10-Q documents across more years, but has a narrower label set.

with less frequent existence ("tail" labels) while our technique is superior on more frequent labels ("head" labels), which aligns with the design of our study, i.e., focusing on the most frequent XBRL labels (Table 3.9).

Furthermore, Sharma et al. (2023) did also a Hits@k evaluation, just like us, demonstrating the practical use of such a system in a possible recommender system with a human-in-the-loop, with models achieving around 90% Hits@5, suggesting the approach can effectively assist human annotators by significantly reducing the number of candidate tags to review, validating our findings. However, they also noted that the high semantic similarity between many financial tags poses a challenge even for domain experts when choosing from the top-k recommendations.

Model (Masking)	Macro-Precision	Macro-Recall	Macro-F1	Micro-Precision	Micro-Recall	Micro-F1
AttentionXML Pipeline ([MASK])	49.83	47.99	46.58	73.91	74.37	74.14
AttentionXML Pipeline ([NUM])	49.01	48.25	46.49	73.57	74.03	73.80
AttentionXML Pipeline ([SHAPE])	**50.69**	**48.51**	**47.54**	**74.50**	**74.96**	**74.74**

Table 3.8: Performance evaluation on FNXL dataset, based on Macro and Micro metrics, using the AttentionXML Pipeline. Results copied from the paper of Sharma et al. (2023).

Model	Macro-Precision	Macro-Recall	Macro-F1
Top 100 frequently occurring labels			
Loukas et al. (2022) avg. (ours)	**90.28**	**77.94**	**82.52**
AttentionXML Pipeline avg.	88.81	77.87	81.97
Least 1000 frequent labels			
Loukas et al. (2022) avg. (ours)	42.60	37.78	39.17
AttentionXML Pipeline avg.	**45.51**	**40.91**	**42.25**

Table 3.9: Bucket analysis for benchmarked techniques on FNXL dataset. Both ours (Loukas et al., 2022) and AttentionXML pipelines incorporate all of our masking techniques and the average is reported. The reported result using our methodology leverages the BERT-base models (and not SEC-BERT). Results copied by the paper of Sharma et al. (2023).

3.14 Can modern LLMs solve this task through prompting?

Xie et al. (2024) created FinBen, an open-source benchmark designed to evaluate large language models (LLMs) on financial tasks. It was developed by the FinAI organization[17], consisting of multiple researchers from Wuhan University, the University of Manchester, Columbia University, and others. FinBen comprises 42 datasets covering 24 distinct financial tasks. These tasks span eight critical aspects like information extraction, textual analysis, question answering, text generation, risk management, forecasting, decision-making, and, bilingual tasks.

In the "information extraction aspect", the authors incorporated the above-mentioned dataset of Sharma et al. (2023) ("FNXL"), which was motivated by our original work on FiNER-139. Then, they benchmarked different Large Language Models on this task using a basic classification prompt that provides all the labels in a list and asks the model to select the appropriate one. As seen in Table 3.10, all LLMs (proprietary and open-weight) fail in almost all test cases of FNXL, with the best score being 1%. The authors note that " [..] for more complex information extraction tasks, such as causal detection (CD) and numerical understanding (FNXL and FSRL[18]), even GPT-4's performance is limited".

We hypothesize that the reason behind LLMs failing almost all of the test cases in financial tagging is due to the problem's very domain-specific nature and the vast amount of labels in such tasks. As noted previously, the semantic similarity between the financial tags in such an extreme classification scenario is so high that we believe only true XBRL experts could separate between those. Notably, only the GPT-4 model scores 1% on the benchmark, while other proprietary models like Google's Gemini 1.0 fail in all test cases. The same happens with other open models of different sizes like Meta's LLama2-70B, LLaMA3-8B, Mistral-7b and other fine-tuned LLMs for finance. This suggests that a different approach should be proposed for solving such numeric entity recognition tasks.

Dataset	Metric	GPT-3.5-turbo	GPT-4	Gemini 1.0	LLaMA2-70B	LLaMA3-8B	FinMA-7B	Mistral-7B
FNXL	EntityF1	0.00	0.01	0.00	0.00	0.00	0.00	0.00

Table 3.10: The zero-shot performance of different LLMs on FNXL, according to the FinBen paper (Xie et al., 2024). All results are the average of three runs.

[17] https://thefin.ai/

[18] FSRL is a textual analogy parsing dataset by Lamm et al. (2018) for extracting and modeling the relationship between analogous facts in financial text.

3.15 Do the masking strategies affect LLMs?

Intrigued by the disappointing results of the LLMs on the FinBen benchmark (Xie et al., 2024) (evaluated on the FNXL dataset), we decided to benchmark some of the latest large language models on our own dataset (FiNER-139). First, we tested how the latest Gemma 3 models by Google (Kamath et al., 2025) performed in a small scale. More specifically, we tested the 4b (*gemma-3-4b-it*) and 27b variants *(gemma-3-27b-it)* by taking the original prompt used by Xie et al. (2024) (Fig. 3.6) and modifying it slightly (Fig. 3.7).

Interestingly, we saw that the small Gemma3 model (4b variant) was not able to follow the instructions correctly and it even sometimes returned random and non-natural characters (artifacts like "Ã©" which are not expected in common language output) instead of the labels instructed, so we did not continue experimenting with this model. The bigger variant model Gemma3-27b was better at following the instructions and indeed returned XBRL labels, but sometimes it was returning the same XBRL labels for all the tokens in a sentence, including obviously non numeric-tokens like "," or "and".

> *In the task of Financial Numeric Extreme Labelling (FNXL),*
> *your job is to identify and label the semantic role of each token in a sentence.*
> *The labels can include {category}.*

Figure 3.6: Original LLM prompt used by Xie et al. (2024) to evaluate various models on the FNXL dataset (Sharma et al., 2023).

> *You are an expert in XBRL classification. Your job is to identify and label each token in a sentence as an XBRL label.*
> *The labels can include {category}.*

Figure 3.7: Our initial, slightly modified prompt inspired by Xie et al. (2024), used in small-scale tests.

After these results, we decided to a) choose a more powerful model (Anthropic's Claude Sonnet 4[19]) and b) give some more context about the task in the LLM instructions, as well as some examples of the task, in order to help the LLM. Finally, we constructed a new (context-enhanced) prompt (Fig. 3.8) for this powerful and proprietary LLM.

[19] https://www.anthropic.com/news/claude-4

You are an expert in XBRL classification in financial documents. You will be given a sentence and you need to classify each token in the sentence with the appropriate label from the provided list of labels. Here are the possible labels you need to assign (IOB2 format):

<LABELS>
<label1, label2, etc.>
</LABELS>

The 'O' tag means that the token doesn't belong to any financial category. All the others indicate that the token belongs to a specific financial category.

Here is also an example of XBRL classification:
For example, if you see the sentence:
'As of December 12 , 2020 , 18 % and 12 % of the outstanding balance of the loan and investment portfolio had underlying properties in New Jersey and Vinewood , respectively !'

The correct IOB2 tags for these tokens (which you should output) are:
O O O O O O O B-ConcentrationRiskPercentage1 O O B-ConcentrationRiskPercentage1 O O O O O O O O O O O O O O O O O O O O O O O O O O O

Return ONLY the IOB2 tags, separated by spaces, with one tag for each token/word, just like the examples.
Do NOT include any other text, explanations, or formatting.

Make sure you do not forget any token, otherwise you will be penalized.
Also, make sure you do not add any extra predictions.

Keep in mind that most tokens are "O" and few of them (mostly some numbers) will have a specific tag from the above ones.

Figure 3.8: Our enhanced LLM prompt used with Anthropic's Claude Sonnet 4 on FiNER-139.

While the results were weak with a 8.31% μ-F_1 and a 7.88% m-F_1, especially for the computational resources of a state-of-the-art LLM, the model, using a proper instruction, managed to predict correct at least some of the test samples, unlike the totally poor outcomes of Xie et al. (2024), which had a max score of 0.1%. This, once again, indicates the difficulty of the real-world XBRL Tagging task. Then, we proceeded with incorporating the [NUM] and [SHAPE] methodologies in Claude Sonnet 4, as introduced in Section 3.9, to test if the masking strategies help the LLMs. Now, since everything was performed at inference time (without any training/fine-tuning), we simply changed the instruction prompt, as seen in Figure 3.9. The masking techniques proved beneficial for the LLM. Using the [NUM] methodology, we had an 1.3% improvement in μ-F_1 (9.64% μ-F_1). While using the [SHAPE] methodology, we

saw an improvement of 2.3%, bringing up the μ-F_1 to 10.6%. While both [NUM] and [SHAPE] improved LLM performance and validated our findings once again, the absolute scores remain low and way lower than using other methodologies (Section 3.6). With 139 entity labels and a highly domain-specific focus of XBRL tagging inside financial statements, this real-world task appears to be particularly challenging even for one of the most powerful LLMs nowadays.

	μ-F_1	m-F_1
Claude Sonnet 4	8.31%	7.88%
Claude Sonnet 4 + [NUM]	9.64%	9.61%
Claude Sonnet 4 + [SHAPE]	**10.60%**	**10.35%**

Table 3.11: Entity-level μ-F_1 and m-F_1 on test data for Anthropic's Claude Sonnet 4 models, along with the addition of [NUM] and [SHAPE] tokens during inference.

Instructions appended to the LLM, when doing inference using the [NUM] methodology

IMPORTANT: In this task, numeric tokens containing numerical digits have been replaced with <NUMBER> using regular expressions..

Instructions appended to the LLM, when doing inference using the [SHAPE] methodology

IMPORTANT: In this task, numeric tokens have been replaced with shape-preserving format:
- Simple numbers like `430` become `[XXX]`
- Complex numbers like `-32.69` become `[-XX.XX]`
- Numbers with commas like `827,404,321` become `[XXX,XXX,XXX]`
- Currency like `$1,234` becomes `[$X,XXX]`
- Percentages like `12.5%` become `[XX.X%]`

Figure 3.9: Instructions appended to our (enhanced) prompt during LLM inference, using the [NUM] and [SHAPE] methodologies in the FiNER-139 experiments. The enhanced prompt can be found in Figure 3.8. When using [NUM] and [SHAPE], we also modified the 1-shot example we showed to the LLM to use the corresponding masking tokens instead of the original numeric ones.

3.16 Conclusion

In this chapter, we introduced XBRL tagging, a real-world financial NLP task driven by regulatory requirements. We released FiNER-139, a large dataset comprising 1.1 million sentences annotated with 139 XBRL tags. We highlighted the unique challenges of this task compared

to standard entity extraction: a significantly larger label set, the prevalence of numeric tokens requiring tags, and the critical role of context over the token itself for correct labelling.

Our experiments compared several neural approaches. We found that standard BERT models struggled due to excessive fragmentation of numeric tokens, leading to a BILSTM model performing better initially. To address this, we proposed simple yet effective method of pseudo-tokens ([NUM] and [SHAPE]) that encode the morphological expressions and the magnitude of the numbers. Those can represent numeric expressions without overfragmentation, significantly improving BERT's performance. These pseudo-tokens also boosted the performance of the domain-specific FIN-BERT model. We further pre-trained and released SEC-BERT, a family of BERT models trained on financial filings on EDGAR-CORPUS, as described in Section 2. When combined with our pseudo-tokens, SEC-BERT achieved the best results on FiNER-139, surpassing prior methods and avoiding the need for computationally expensive CRF layers. Surprisingly, we also demonstrated that LLMs struggle significantly on this real-world task, which involves a large number of entity labels within a highly specialized domain. Our proposed tokenization/masking techniques provided a modest improvement to the LLMs too, validating our findings.

Our work on FiNER-139 directly motivated subsequent research, notably the FNXL project (Sharma et al., 2023) from Goldman Sachs, which scaled the task to nearly 2,800 labels. The FNXL task was later incorporated into FinBen (Xie et al., 2024), the state-of-the-art benchmark for financial NLP. Results from both the FinBen paper (Xie et al., 2024), as well as our follow-up experiments, indicate that even modern large language models like GPT-4 or Claude Sonnet 4 find the extreme classification nature of this task, with its vast number of potential tags, highly challenging and fail nearly at all or most test cases, which highlights the need for alternative approaches in order to achieve full automation.

Chapter 4

Resource-Limited Intent Recognition in Banking

4.1 Introduction

In this chapter, we study a resource-limited scenario of intent recognition, a form of text classification where the goal is to infer the underlying user intention from short utterances, typically in customer dialogues. This task is critical in many industrial applications, such as virtual assistants and banking support systems, where correctly identifying user intent affects the next state of a multi-state pipeline. However, intent recognition often faces resource-limited conditions in SMEs, both in terms of data scarcity and deployment cost.

Natural Language Processing (NLP) has advanced significantly, especially in text classification. Traditional (full-data) classifiers need thousands of labeled examples, which is often not practical in data-limited fields like finance (Casanueva et al., 2020). Modern Few-Shot methods, such as PET (Schick and Schütze, 2021) and SetFit (Tunstall et al., 2022) aim to address this by requiring only 10 to 20 examples per class in BERT-based models. Recent work also focuses on prompting Large Language Models (LLMs) like GPT-3.5 and GPT-4 with just 1-5 examples per class, often through cloud-based commercial APIs. However, the tradeoffs between the performance and operating cost (OpEx) of these methods are underexplored.[1]

Specifically, in this work, we bridge this gap by evaluating different approaches in few-shot scenarios using the real-world Banking77 financial intent recognition dataset (Casanueva

[1]This research was conducted during my work at helvia.ai, a start-up with conversational AI assistants, where we needed to find the best balance between performance and cost efficiency for a text classification project we wanted to deploy for a customer in the banking industry.

et al., 2020).[2] We include evaluations of modern LLMs from OpenAI, Cohere, and Anthropic. Banking77 is a real-world dataset of customer service intents. Unlike other intent datasets, it has many (77) labels with semantic similarities (Table 4.1), making it suitable for studying methods while addressing a practical business case.

Intent	**Label**
My card was declined.	Declined Card Payment
It declined my transfer.	Declined Transfer
How do you calculate your exchange rates?	Exchange Rate
My card was eaten by the cash machine.	Card Swallowed
I lost my card in the ATM.	Card Swallowed
I got married, I need to change my name.	Edit Personal Details
...	...
My card is needed soon.	Card Delivery Estimate
What is the youngest age for an account?	Age Limit
What caused my transfer to fail?	Failed Transfer
I have noticed that my account was double charged	Transaction Charged Twice

Table 4.1: Example banking intents and their labels from the Banking77 dataset. In total, there are 77 different labels in the dataset with highly overlapping semantic similarities.

4.2 Contributions

The key contributions of this chapter's work are:

1. We provide a comprehensive study benchmarking and comparing multiple approaches for intent recognition on the Banking77 dataset, with a primary focus on Few-Shot learning:

    - Fine-tuning a BERT-based masked-language model, namely MPNet (Song et al., 2020), in a Full-Data setting where the model is trained on the entire training dataset of approximately 10k examples.
    - Few-shot learning using SetFit (Tunstall et al., 2022) on MPNet with 20 or fewer examples per class.

[2] Our findings are also later validated by other studies in more datasets.

- Few-shot learning using in-context learning with popular LLMs including OpenAI GPT-3.5-turbo, OpenAI GPT-4, Cohere Command-nightly, and Anthropic Claude 2, using only 1 and 3 examples per class.

2. We demonstrate that in-context learning with LLMs can outperform BERT-based models in this financial few-shot task when not enough data can be provided (in our scenario, 3 samples per class or less). However, interestingly, we show that if more data points per class are provided (5 and more) by a domain expert, BERT-based models with SetFit can outperform commercial API endpoints from LLM providers, while they are also significantly cheaper to deploy.[3] This is important since in real-world settings, a customer has limited time for collaboration, and while asking them for up to 5 samples per class is relatively achievable, collecting the extensive labeled datasets required by traditional machine learning approaches is often impractical.

3. We show that using expert-curated examples for in-context learning yields substantial performance gains (up to 10 points) compared to randomly selected examples.

4. We also investigate replacing the expert-curated samples for Few-Shot with synthetic data augmentation using GPT-4 in a simulated low-resource scenario, demonstrating still an improved performance, but eliminating any need for manual annotation effort by the customer/domain-expert.

5. We analyze the operational costs (OpEx) associated with cloud-based LLM API providers across different few-shot techniques for intent recognition.

6. We investigate and demonstrate "Dynamic Few-Shot Prompting", a cost-effective LLM inference method utilizing Retrieval-Augmented Generation (RAG) that selects the most similar training samples during inference. This technique is inspired by older studies (Liu et al., 2022) that focused solely on improving the performance scores. We show that this approach can notably cut API costs (e.g., $535 savings with 5 retrieved examples, achieving 84.5% vs. 83.1% for standard 3-shot at $740) while maintaining state-of-the-art performance by retrieving just a small portion (2.2%) of examples.

To the best of our knowledge, this is the first study that presents together a comprehensive evaluation of such methods in both a data-limited and cost-limited industrial context, with

[3]This research was conducted during June 2023, three months after the API release of OpenAI's models.

particular emphasis on the operational costs of commercial LLM API providers where pricing scales with token usage.

4.3 Related Work

Studies on Banking77

Casanueva et al. (2020) introduced the Banking77 dataset, achieving baseline accuracies of 93.6% in the Full-Data setting and 85.1% in a 10-shot setting by fine-tuning BERT (Devlin et al., 2019b) and using the Universal Sentence Encoder (Cer et al., 2018). The Universal Sentence Encoder is a pre-trained Transformer model, trained via multi-task learning on large-scale text corpora, that maps entire sentences into fixed-length embeddings, capturing sentence-level semantics and enabling efficient downstream classification with minimal task-specific supervision. Later, Ying and Thomas (2022) improved the dataset quality by identifying and correcting mislabeled examples using confident learning (Northcutt et al., 2021), a method that estimates the joint distribution of noisy and true labels to detect likely annotation errors. This led to improved training reliability and a better Full-Data accuracy of 92.4%.

Li et al. (2022) proposed a parameter-efficient transfer learning method for intent recognition by pre-training intent representations through two main techniques. First, they applied prefix-tuning, wherein trainable prefix vectors are prepended to each Transformer layer's key and value projections. These prefixes act as soft prompts that steer the model toward learning intent-relevant features. Second, they used partial fine-tuning, updating only the final Transformer layer of BERT while keeping the rest of the model frozen. Using this approach, they improved performance on the Full-Data benchmark.

Additionally, Mehri and Eric (2021) introduced two complementary ideas to enhance few-shot intent recognition: example-driven training and observers. Their example-driven training framework avoids parametric softmax classification and instead reframes intent prediction as a non-parametric similarity task. A BERT-style encoder computes vector representations for test utterances and labeled support examples, and classification is done by measuring similarity to the most relevant support examples per class. This approach allows the model to generalize better to low-resource settings. To further improve sentence-level representations, they proposed "observers", some special tokens that attend to the input sequence but do not receive attention from it. By averaging the embeddings of these unidirectional observer tokens at the final layer, the model obtains more robust utterance representations than those derived

from the standard [CLS] token. With these techniques, their model achieves 85.9% accuracy in the 10-shot setting and 93.8% using the full dataset.

Lastly, Zawbaa et al. (2024) introduced DETER, a dual-encoder architecture tailored for intent recognition and robust out-of-scope (OOS) detection in dialogue data sets such as Banking77 (Casanueva et al., 2020). The introduced model combines the Universal Sentence Encoder (Cer et al., 2018) with a Transformer-based denoising autoencoder to encode user utterances. It incorporates a threshold-based re-classification mechanism, which refines initial predictions by tagging an utterance as OOS if its confidence score falls below a pre-computed threshold. It also augments the training set with synthetically generated outlier examples to improve generalization. In their setup, "known intents" refer to classes included in the training set (e.g., 25% or 50% of the 77 Banking77 intent categories), while "out-of-scope intents" correspond to user queries that fall outside the trained intent distribution and must be flagged as unknown. Despite being trained with limited intent coverage, DETER achieved strong results, with macro F1 scores exceeding 87% on known intents and 93% on OOS detection in their unique setup.

Few-Shot Text Classification

Learning from only a few training examples is vital, especially in real-world use cases where there are no prior datasets and typically there are limited or no resources to create one.[4] In such cases with very limited data, fine-tuning often performs poorly (Dodge et al., 2020) and actually becomes more challenging as language models grow in size. An alternative approach is in-context learning (Brown et al., 2020), which involves prompting a (generative) Large Language Model(LLM) with a context and asking it to complete NLP tasks without fine-tuning. The context usually includes a brief task description, some examples (the context), and the instance to be classified. The intuition behind in-context learning is that the language model has already learned several similar tasks during pre-training, and the prompt attempts to identify the appropriate one (Reynolds and McDonell, 2021). To better align Large Language Models (LLMs) with user intent, researchers have fine-tuned these models to explicitly follow human instructions (OpenAI, 2022, 2023). While this instruction tuning has led to noticeable improvements, in-context learning can still underperform in certain settings (Razeghi et al., 2022).

[4]For example, when working in the industry, customers rarely have the time to provide you with unlabeled or even labeled data. This was also happening in our case.

4.4 Task and Dataset

Intent recognition is a specific instance of text classification and a vital component of task-oriented conversational systems across multiple fields, such as finance. Real-world commercial intent recognition systems are often complex due to the nature of their intent categories. These categories are typically fine-grained and partially overlapping, making classification more challenging. The situation is further complicated by the lack of comprehensive datasets in domains like finance (Casanueva et al., 2020; Loukas et al., 2021b, 2022; Zavitsanos et al., 2022).

Publicly available intent recognition datasets are also limited in scope, and the most commonly used ones are overly generic, failing to capture the complexity of real-world industrial systems like the one studied in our work (Braun et al., 2017; Coucke et al., 2018). In response to the need for industry-ready datasets (Larson et al., 2019; Liu et al., 2021), PolyAI released Banking77 (Casanueva et al., 2020), which focuses on a single domain (banking) and consists of 77 domain-specific intents. By concentrating on a specific domain and offering a diverse set of intents, the dataset emulates a more realistic and challenging intent recognition task than most generic benchmarks. Also, it is worth noting that some intent categories partially overlap, requiring fine-grained labels that cannot rely solely on the semantics of individual words, further highlighting the task's difficulty.

The Banking77 dataset comprises 13,083 annotated customer service queries labeled with 77 intents and is split into two subsets: train (10,003 examples) and test (3,080 samples) (Table 4.2). The label distribution is heavily imbalanced in the training subset (Figure 4.1), demonstrating the challenge of developing classifiers in the Full-Data setting. The test subset comprises 40 instances for every label.

Banking77 Statistics	**Train**	**Test**
Number of examples	10,003	3,080
Average length (in characters)	59.5 ± 40.9	54.2 ± 34.7
Average length (in words)	11.9 ± 7.9	10.9 ± 6.7
Number of intents (classes)	77	77

Table 4.2: Banking77 dataset statistics. The average lengths are shown along with their corresponding standard deviations.

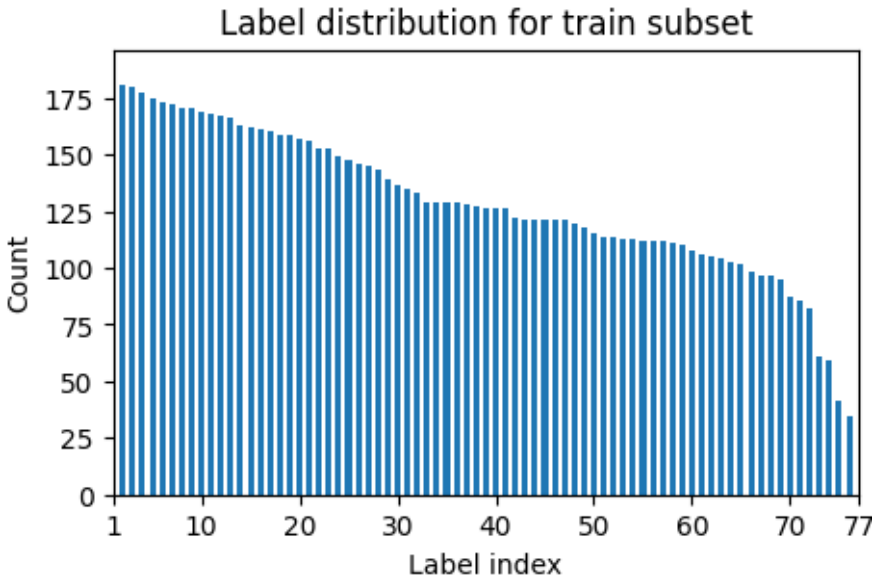


Figure 4.1: Class distribution of the 77 intent labels found in the training set of Banking77. Intent indices are shown instead of tag names for brevity.

4.5 Methodology

4.5.1 Fine-tuning BERT-based models

MPNet (Song et al., 2020) is a transformer-based model (Vaswani et al., 2017; Devlin et al., 2019b) that introduces improvements in its pre-training strategy to better capture token dependencies within a sequence. One of its main innovations is the use of *permuted language modeling*, inspired by XLNet (Yang et al., 2019). Unlike BERT's masked language modeling, where random tokens are masked and predicted, permuted language modeling trains the model to predict tokens based on multiple possible orderings of the input. MPNet combines this permutation-based objective with masked prediction. This allows it to capture both bidirectional context and flexible dependency structures, leading to richer contextual representations and better performance.

Like other transformer models, MPNet uses learned absolute positional embeddings similar to BERT at both pre-training and inference. MPNet's distinguishing feature is that during pre-training, it combines masked language modeling with permuted language modeling, where masked tokens are predicted in a randomized factorization order rather than left-to-right or bidirectionally. During this permuted prediction process, the model learns to attend to tokens across different positions in various orders, which helps it better capture dependencies between words based on their context and not just their absolute or relative sequential

positions. This training approach improves the model's understanding of contextual relationships without requiring architectural changes to the positional encoding scheme itself.

MPNet was pre-trained on a 160GB English text corpus and consistently outperforms earlier models such as BERT, XLNet, and RoBERTa (Liu et al., 2019) across a range of natural language understanding tasks. In our experiments, we use `all-mpnet-base-v2`[5], which is among the top-performing models in the Sentence Transformers leaderboard[6] which we used at the time of the study[7], making it a prominent choice for our task.

4.5.2 Few-Shot Contrastive Learning with SetFit

SetFit (Tunstall et al., 2022) is a methodology developed by HuggingFace. It enables efficient fine-tuning of Sentence Transformer models using only a small number of labeled examples per class.[8] SetFit is particularly suited for low-resource classification settings, combining contrastive representation learning with a simple classification head trained on learned embeddings.

SetFit's training pipeline consists of two stages (Figure 4.3). First, it fine-tunes a pretrained Sentence Transformer using contrastive learning in a Siamese network setup **??**. In this stage, the model is presented with sentence pairs: positive pairs consist of two examples from the same class, while negative pairs come from different classes. The model is optimized using a contrastive loss (e.g., cosine similarity loss), encouraging the embeddings of positive pairs to be close and those of negative pairs to be distant. This leads the encoder to learn semantically meaningful and class-discriminative representations, even when labeled data is scarce.

SetFit can significantly increase the efficiency of the available data by generating all possible pairs of examples. Given only a few labeled examples per class, it constructs all possible in-class and cross-class sentence pairs, creating a much larger and more expressive training signal from limited data.

In the second stage, the transformer (which has been trained with contrastive learning) is used to embed the original sentences individually. A simple classifier, typically a linear layer, is then trained on these embeddings to perform the actual classification. Crucially,

[5] `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`

[6] `https://www.sbert.net/docs/pretrained_models.html`

[7] Nowadays, more difficult benchmarks exist for sentence embeddings, like MTEB (Muennighoff et al., 2023): `https://huggingface.co/spaces/mteb/leaderboard`

[8] `https://github.com/huggingface/setfit`

this separation between representation learning and classification allows the transformer to generalize better from few-shot data without overfitting to specific class labels during fine-tuning.

Despite using as few as 8 examples per class in some setups, SetFit achieves performance comparable to or better than conventional fine-tuning approaches that rely on full training sets (Tunstall et al., 2022). This makes it a powerful alternative in real-world settings with limited labeled data, such as intent recognition or domain-specific text categorization.

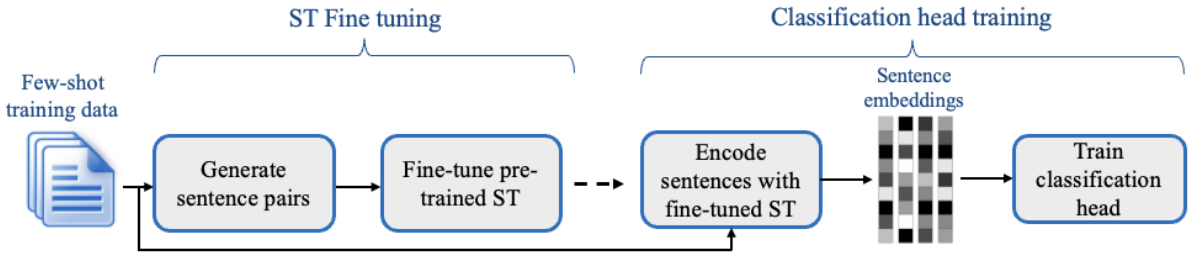


Figure 4.2: A diagram of SetFit's two-stage training process. Retrieved from HuggingFace's blogpost: `https://huggingface.co/blog/setfit`.

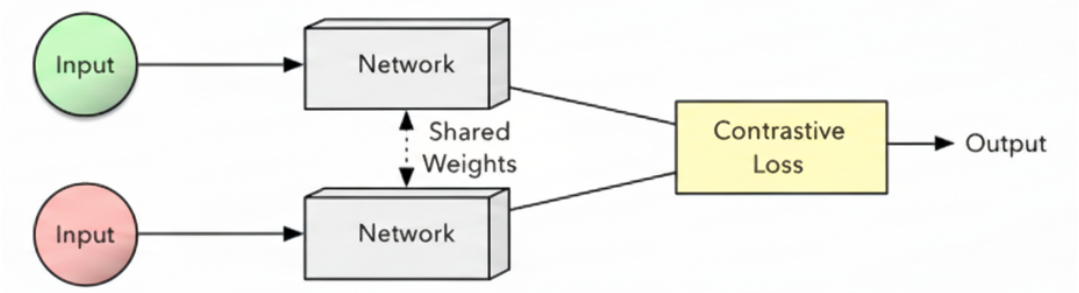


Figure 4.3: In Stage 1, SetFit uses a Siamese network architecture to fine-tune sentence transformers on labeled pairs using contrastive loss, creating task-specific embeddings for the subsequent classification stage.

4.5.3 In-Context Learning

For in-context learning, we use closed-source LLMs such as **GPT-3.5-turbo** (OpenAI, 2022) and **GPT-4** (OpenAI, 2023), which are based on the Generative Pre-trained Transformer (GPT) architecture (Radford and Narasimhan, 2018; Radford et al., 2019). These pre-trained models are first fine-tuned with instruction-tuning, where they learn to follow human instructions by training on large datasets of input-output pairs (human requests and ideal responses).

Subsequently, Reinforcement learning from human feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022) is applied to further refine model behavior and align outputs with human preferences. At the time of our research, GPT-3.5-turbo was available in two context window variants, supporting 4,096 and 16,384 tokens respectively. GPT-4 is a multimodal model that was also served in two variants with context windows of 8,192 and 32,768 tokens. Additionally, we used **Cohere's Command-nightly** with a 4,096-token context window and **Anthropic Claude 2**, which supported a larger context window of 100,000 tokens.[9]

4.5.4 Human Expert Annotation for Robustness

Class overlaps were discovered by Casanueva et al. (2020) during the creation of Banking77. To address this issue, previous studies like Ying and Thomas (2022) used additional annotation to curate subsets and enhance their performance. However, they did not share their curated subset for reproducibility.

To tackle these such problems with the overlaps, we curated a subset of Banking77 by hiring a subject matter expert. We provided the human expert with 10, randomly-picked examples per class, and they selected the top 3 based on their alignment with the corresponding intents. This meticulous approach allowed us to produce a subset with reduced class overlaps and ensured the high relevance of each example to its intended label. As we show later, this is fundamental in the few-shot scenario since this subset, containing these expert-selected training instances, outperformed randomly selected instances per class when incorporating them in language models. To encourage more NLP research, we provide this curated subset as a free resource for the community.[10]

4.6 Experiments & Results

4.6.1 Experimental Setup

For the **fine-tuning** methods, we use TensorFlow and HuggingFace. We deployed all the BERT-based (MLM) pipelines in an NVIDIA A100 SXM GPU with 40GB VRAM. For all **in-context learning** methods, we use the official APIs from the corresponding LLM vendors

[9] See `https://cohere.com/` and `https://www.anthropic.com/`. We used the Command-nightly version available during the week of 2–9 October 2023. Anthropic does not specify checkpoint details for Claude 2.

[10] The curated subset is hosted at `https://huggingface.co/datasets/helvia/banking77-representative-samples`.

and we instruct the model to return only the financial intent label of the test example that is presented in each query. The prompt we use can be broken down into 3 parts and is the same for all LLMs, i.e., GPT-3.5-turbo and GPT-4 (OpenAI),[11] Command (Cohere), and Claude (Anthropic). The first part contains the description of the task and the available classes. The second provides a few examples, and the third presents the text to be classified. The prompt template is shown in Figure 4.4.

```
You are an expert assistant in the field of customer service.
Your task is to help workers in the customer service
department of a company. Your task is to classify the
customer's question in  order to help the customer service
worker to answer the question.

In order to help the worker, you MUST respond
with the number and the name of one of the
following classes you know. If you cannot answer the question,
respond: "-1 Unknown".

In case you reply with something else, you will be penalized.

The classes are:
0 activate_my_card
1 age_limit
2 apple_pay_or_google_pay
3 atm_support
..       ..
75 wrong_amount_of_cash_received
76 wrong_exchange_rate_for_cash_withdrawal

Here are some examples of questions and their classes:
How do I top-up while traveling? automatic_top_up
How do I set up auto top-up? automatic_top_up
...               ...
It declined my transfer. declined_transfer
```

Figure 4.4: The prompt template we used to query the Large Language Models (LLMs).

4.6.2 Hyperparameter tuning in BERT-based models

We tuned the hyper-parameters of MPNet-v2 using Optuna's (Akiba et al., 2019) implementation of the Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al., 2011). We specified 10 trials, and we defined a search space of (1e-5, 5e-5) for the body's learning rate

[11] We use the available `gpt-3.5-turbo` variants of 4K and 16K contexts for the 1-shot and 3-shot settings accordingly, and the `gpt-4` 8K context model, which is less expensive than the 32K one. In all cases, we use the `0613` checkpoints.

and (1e-2, 5e-5) for the head's learning rate with logarithmic intervals. During tuning, we maximized the validation Micro-F1.

4.6.3 Results

To evaluate model performance, we report both micro-F1 (μ-F_1) and macro-F1 (m-F_1) scores. Micro-F1 is computed by summing the true positives, false positives, and false negatives across all intent classes and then calculating the F1 score from these totals, giving more weight to frequent intents. Macro-F1 computes the F1 score independently for each intent and then averages them, treating all intents equally regardless of their frequency.

Competitive results are achieved by MPNet-v2 across all few-shot settings using SetFit, as shown in Table 4.3. When trained on only 3 samples, it achieves scores of 76.7 μ-F_1 and 75.9 m-F_1. As we increase the number of samples, the performance improves, reaching a 91.2 micro-F1 and 91.3 macro-F1 score with 20 samples. This is only 3 percentage points (pp) lower than fine-tuning the model with all the data, demonstrating the effectiveness of SetFit, especially in domains where acquiring data points is difficult. Lastly, MPNet-v2 surpasses the results reported by Mehri and Eric (2021), outperforming them by 2.2 percentage points in the 10-shot setting using SetFit, and by 0.3 percentage points in the Full-Data setting with traditional fine-tuning.

Despite being presented with only 1 sample per class (either random or representative), GPT-4 achieves competitive results (80.4 and 78.1 μ-F_1). It outperforms MPNet-v2 by a large margin (over 20 pp) in the 1-shot setting, showing the potential for effective few-shot classification in domains where data is limited (Loukas et al., 2021b).

Better in-context learning results (both in GPT-3.5-turbo and GPT-4) are achieved using our human-curated representative samples. We also employed two alternative closed-source LLMs by Cohere and Anthropic on the representative samples to see how they stand against the LLMs by OpenAI. Cohere's Command-nightly performs poorly with a low 58.4 μ-F_1 while Anthropic's Claude 1 yields a 73.8 μ-F_1, comparable to GPT-3.5's 75.2. At the same time, Anthropic's Claude 2 performs slightly better with a 76.8 μ-F_1.

We proceed with 3-shot classification with the top-performing model, GPT-4, and the less powerful GPT-3.5-turbo model of the OpenAI models family. GPT-4 outperforms both previous models on the 3-shot setting by more than 6 pp (MPNet-v2) and 17 pp (GPT-3.5). More notably, GPT-3.5-turbo, performs poorly on its 3-shot variant (65.5 μ-F_1) compared to its promising 1-shot variant (75.2 μ-F_1). This probably verifies recent reports on GPT-

3.5-turbo getting lost in the middle of long contexts (Liu et al., 2023). Similarly to the 1-shot experiments, GPT-4's performance drops substantially (approximately 9 pp) when shown random samples instead of representative ones. Thus, it is better to present more examples to a more powerful engine like GPT-4 instead of GPT-3.5-turbo.

Methods	Setting	μ-F_1	m-F_1
Mehri and Eric (2021)	Full-Data	93.8	NA
Mehri and Eric (2021)	10-shot x 77	85.9	NA
Ying and Thomas (2022)	Full-Data	NA	92.0
MPNet-v2	Full-Data	**94.1**	**94.1**
MPNet-v2 (SetFit)	1-shot x 77	**57.4**	**55.9**
GPT-3.5 (representative samples)	1-shot x 77	**75.2**	**74.3**
GPT-3.5 (random samples)	1-shot x 77	74.0	72.3
GPT-4 (representative samples)	1-shot x 77	**80.4**	**78.1**
GPT-4 (random samples)	1-shot x 77	77.6	76.7
Command-nightly (representative samples)	1-shot x 77	58.4	57.8
Anthropic Claude 1 (representative samples)	1-shot x 77	73.8	72.1
Anthropic Claude 2 (representative samples)	1-shot x 77	76.8	75.1
MPNet-v2 (SetFit)	3-shot x 77	**76.7**	**75.9**
GPT-3.5 (random samples)	3-shot x 77	57.9	59.8
GPT-3.5 (representative samples)	3-shot x 77	**65.5**	**65.3**
GPT-4 (representative samples)	3-shot x 77	**83.1**	**82.7**
GPT-4 (random samples)	3-shot x 77	74.2	73.7
MPNet-v2 (SetFit)	5-shot x 77	**83.5**	**83.3**
MPNet-v2 (SetFit)	10-shot x 77	88.1	88.1
MPNet-v2 (SetFit)	15-shot x 77	90.6	90.5
MPNet-v2 (SetFit)	20-shot x 77	**91.2**	**91.3**

Table 4.3: Classification results on the Banking77 test set across all evaluated models. "N-shot $\times$ 77" denotes that N examples per class (77 intent classes in total) were used either for training (in the case of BERT-based models with SetFit) or for prompting (in the case of LLMs). For LLMs, representative samples refer to 3 examples per class selected by a domain expert, while random samples are drawn without any curation. The MPNet-v2 model is also evaluated after full fine-tuning (without SetFit) on the complete dataset (Full-Data setting) for comparison.

4.7 Cost-Effective LLM Inference using RAG

4.7.1 Cost Analysis

Apart from studying the models' performance, we investigate the significant Operating Costs (OpEx) associated with popular LLMs like GPT-4. We provide a budget analysis for our experiments in Table 4.4 (cost refers to ∼3k instances). This can be seen as a guideline for researchers and practitioners to evaluate the trade-offs between performance and budget when selecting a closed-source LLM offered via a commercial API, as well as a data point when deciding between a "build vs. buy" approach, where building requires developing datasets and setting up and hosting a custom model.

The **two upper groups** of the table illustrate how popular LLMs perform in the financial intent recognition task of Banking77 in a standard few-shot approach. We observe that in the 1-shot setting, Anthropic Claude 2 has a performance that comes on par with the GPT-3.5 model and comes at half of GPT-3.5's cost. In the 3-shot setting, GPT-4 scores nearly 20 points more than GPT-3.5, but comes with a 10x cost, which amounts to $740.

4.7.2 Dynamic Few-Shot Prompting with Retrieval-Augmented Generation (RAG)

Motivated by the high costs associated with closed-source LLMs (Stogiannidis et al., 2023), we present a methodology to query LLMs efficiently by creating a Retriever component before plugging it into our Generative Text Classifier (Reader) pipeline.

The standard Few-Shot setting involves providing examples for all intents; e.g., in the 3-shot setting, we use a prompt with 231 samples (3 samples x 77 classes) as context in each inference call to the LLMs. However, our intuition is that a small subset of them can be sufficient, allowing room for a cost-effective approach by narrowing down the number of representative examples used in each inference call.

Thus, we consider including in the prompt only the most similar examples (similar to the particular input being classified) following an active learning and nearest-neighbor selection strategy inspired by older studies in text generation models (Lewis et al., 2020; Liu et al., 2022). The same technique has also been independently explored for Large Language Models in concurrent research research studies (Milios et al., 2023; Bouzaki, 2023) which focused primarily on performance improvements. In contrast, our work uniquely addresses both performance and the operational costs associated with commercial LLM API providers,

Model	Setting	Micro-F1↑	Cost↓
Standard Few-Shot			
GPT-4	1-shot x 77	**80.4**	$620
GPT-3.5	1-shot x 77	75.2	$31
Anthropic Claude 2	1-shot x 77	76.8	**$15**
Command-nightly	1-shot x 77	58.4	$22
GPT-3.5	3-shot x 77	65.5	**$62**
GPT-4	3-shot x 77	**83.1**	$740
Dynamic Few-Shot (RAG)			
GPT-4	5 similar (RAG)	84.5	$205
Anthropic Claude 2	5 similar (RAG)	**84.8**	**$33**
GPT-4	10 similar (RAG)	81.2	$230
Anthropic Claude 2	10 similar (RAG)	**85.2**	**$37**
GPT-4	20 similar (RAG)	**87.7**	$270
Anthropic Claude 2	20 similar (RAG)	85.5	**$42**

Table 4.4: Cost analysis of various closed-source LLMs. The cost is calculated by counting the input tokens sent to the LLM. We do not count output tokens since they consist of few words, are dynamic and they are negligible. The first two groups represent the 1- and 3-shot results, multiplied by 77 classes. The rest groups come from utilizing Dynamic Few-Shot Prompting using Retrieval-Augmented Generation (RAG) for cost-effective LLM inference. Note that in the latter method, we only keep the top K=5/10/20 similar and we do not retrieve other samples. We perform 3,080 queries to each LLM, i.e., 1 query per sample in the test set.

analyzing the cost-efficiency trade-offs that are critical for resource-limited organizations like startups and small research groups. While the performance of similar retrieval-based augmentation has been explored in prior literature using locally-loaded models, we show that its application to commercial LLM APIs, where pricing scales with token usage, has a significant and previously underexplored impact on cost-efficiency. Specifically, instead of always prompting the LLM with all 231 examples, we include only the top 5 (2.2%), 10 (4.3%), or 20 (8.7%) most similar ones per instance, based on cosine similarity using `all-mpnet-base-v2` embeddings (Figure 4.5).

Using this approach, improved results both in performance and budget are shown, as seen in the three lower groups of Table 4.4. For instance, when we retrieve only five similar instances (2.2% of the 231 total examples) for each inference call, Anthropic Claude 2 has an 84.8 μ-F_1 vs. GPT-4's 84.5. At the same time, it costs \$172 less (1/6 of GPT-4's cost). Most importantly, this score is also higher than 1.5 percentage points than GPT-4 in the classic 3-shot setting, which costs \$740 and is 22 times more costly.

By increasing the retrieved similar samples to 10 and 20, we see a slight increase to a maximum of 87.7% μ-F_1 for GPT-4 (270\$) and 85.5% μ-F_1 of Claude 2 (42\$). Anthropic Claude 2 is consistent in improving over the number of retrieved examples. In contrast, GPT-4's performance slightly decreases when moving from 5 to 10 similar examples but increases again when provided with 20 examples. One possible explanation for this is the non-deterministic behavior of LLMs.

To formalize the trade-off between performance and cost, we can also define a cost-efficiency framework. The total cost of LLM inference from a cloud provider (like OpenAI or Google) is primarily a function of the number of input tokens, excluding the use case's task instructions and the (few) LLM-generated output tokens. Thus, in the standard S-shot setting with C classes, the number of example tokens per query is proportional to the product S x C. In contrast, the Dynamic Few-Shot approach using RAG decouples the prompt size from the total number of classes, making the number of example tokens proportional only to K, the number of retrieved samples. The cost reduction can therefore be approximated by the ratio of example tokens, which simplifies to K / (S x C). For the standard 3-shot setting, this means using K=5 in Dynamic Few-Shot Prompting can reduce the LLM example tokens payload to just 5 / (3 x 77) $\approx$ 2.2% of the original, directly translating into the significant cost savings observed in Table 4.4.

This framework allows us to also analyze the optimal value for K in Dynamic Few-Shot Prompting by evaluating the marginal gain in performance against the linear increase

in cost. To quantify this, we can introduce a **Cost-Effectiveness Score (CES)**, defined as CES = Micro-F1/Cost, which represents the performance points gained per dollar. For example, applying this to our GPT-4 results, the standard 3-shot setting yields a CES of $83.1/740 \approx 0.11$, establishing a baseline. In contrast, the dynamic Few-Shot prompting approach with $K = 5$ achieves a dramatically higher CES of $84.5/205 \approx 0.41$, highlighting its efficiency. However, as we increase the number of retrieved examples, we observe a clear pattern of diminishing marginal returns. While increasing K from 10 to 20 improves the Micro-F1 score from 81.2 to 87.7 (an 8.0% increase), the associated cost rises from $230 to $270 (a 17.4% increase). Consequently, the CES for GPT-4 declines from 0.41 ($K = 5$) to 0.35 ($K = 10$) and further to 0.32 ($K = 20$). This demonstrates that each additional example in Dynamic Few-Shot Prompting provides progressively less performance benefit relative to its cost. The optimal choice of K is therefore not the one that maximizes performance in isolation, but the one that best balances the trade-off between performance and the available budget, with smaller K values representing the most efficient use of resources. This is particularly important since most LLMs, in reality, are being used for inference through external cloud providers who charge per token. In practice, resource-constrained startups or research groups can optimize the performance-cost tradeoff by tuning K on a development set to find an affordable configuration before running inference on the test set.

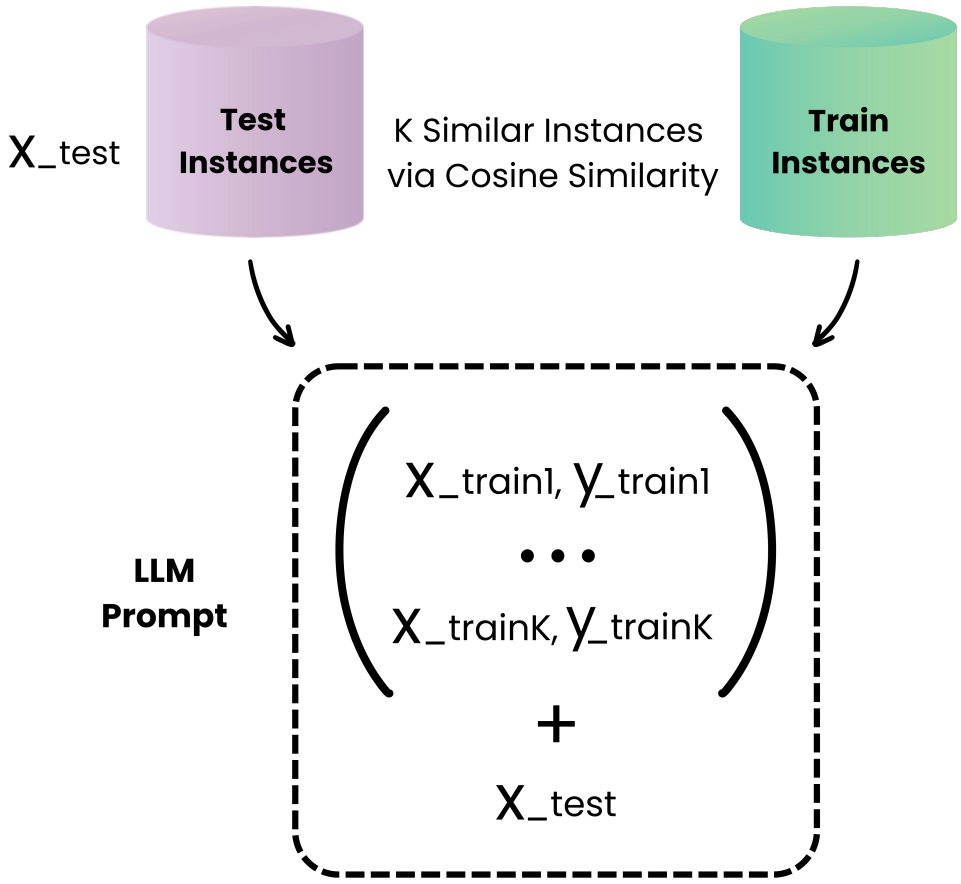


Figure 4.5: Dynamic Few-Shot Prompting: Our LLM prompt is constructed dynamically through Retrieval-Augmented Generation (RAG), using cosine similarity for in-context data selection between the test inference sample and our train samples. We use K=5, 10, 20.

4.8 LLMs for Data Generation in Low-Resource Settings

Data-centric AI[12], which emphasizes improving the quality, representativeness, and usability of data rather than solely focusing on model architectures to enhance machine learning performance, is increasingly gaining attention, as evidenced by relevant scientific venues.[13] Such approaches become invaluable in areas where collecting vast quantities of data is either infeasible, prohibited by privacy constraints, or particularly challenging, such as in finance.[14]

Following this low-resource paradigm, we simulate a scenario where our dataset is limited. Assuming that providing 3 examples per class is feasible in practice, we start with the

[12]https://mitsloan.mit.edu/ideas-made-to-matter/why-its-time-data-centric-artificial-intelligence

[13]https://datacentricai.org/neurips21/

[14]https://sites.google.com/view/icaif-synthetic/home

231 original handpicked examples from the expert (3 examples per each of the 77 classes), and we leverage GPT-4 to generate additional examples per class. The intention here is to explore how much we can rely on data augmentation, prompting LLMs to generate synthetic examples in order to improve performance, and identify the threshold beyond which the quality of generated examples starts to degrade, and thus, we cannot continue with adding more (artificial) examples. While past existing studies show that just writing simple prompts boosts performance on various classifications tasks, they come short due to two reasons: (a) they consider vanilla tasks with up to 6 classes (where in our scenario, we consider 77) (Yoo et al., 2021) and (b) they use older text generation models rather than instruction-tuned LLMs.

Given that Sahu et al. (2022) observed limited or no benefits from data augmentation using an older text generation model without prompt engineering (GPT-3) on tasks with large label sets like Banking77 (77 labels) and HWU64[15] (64 labels), we adopt a more advanced strategy, by feeding the most similar labels (and their sentences) together in groups/buckets to GPT-4, the instruction-tuned model we used.

We begin with a detailed data analysis using Azimuth (Gauthier-melancon et al., 2022), an open-source toolkit for data and error analysis. We group the 77 labels into ten clusters based on semantic similarity by computing BERT (Devlin et al., 2019a) embeddings on the label names. Labels with closely related intents such as Top Up Reverted and Top Up Failed are placed in the same group. For each semantic group, we select 3 example sentences per label and prompt the LLM to generate 20 synthetic examples per label while explicitly emphasizing these semantically overlapping classes (Figure 4.6). This targeted approach encourages the LLM to produce more distinct and informative examples and aims to improve performance beyond the limited gains reported by Sahu et al. (2022).

After having generated some artificial samples, we employed the SetFit approach across four different settings: 5-shot, 10-shot, 15-shot, and 20-shot. That way, we can compare the performance with the original settings (i.e., using SetFit on data from the real dataset). However, when incorporating the generated data, we modified the traditional methodology of presenting the N original examples per class. Instead of the 5-shot experiment, we used 3 examples from the original data and 2 from the synthetic ones. Similarly, the original-to-augmented ratios for 10-shot, 15-shot, and 20-shot tasks were 3:7, 3:12, and 3:17, respectively, i.e., we always used only 3 original examples when using data augmentation. This allowed us to assess how well our model performs with limited data and how effectively it can integrate and learn from artificially generated examples.

[15] https://huggingface.co/datasets/DeepPavlov/hwu64

The μ-F_1 score for different few-shot settings is shown in Figure 4.7. Starting from the 3-shot setting with 76.7% μ-F_1, one can generate artificial data to get an essential boost to 81% μ-F_1 as seen in the augmented 5-shot setting (3 original examples + 2 generated). The maximum performance increase is observed at the 10-shot augmentation scenario with 81.6%. After this threshold, the quality of the data generated decreases (see 15- and 20-shot settings), injecting more noise into the model than the actual value. As expected, the real data (even if they are challenging to obtain in such domains) are far superior and more meaningful than the generated data (91.2 μ-F_1 vs. 78.8 μ-F_1 in the 20-shot setting).

4.9 Error Analysis

Lastly, to understand our models' limitations, we also performed an exploratory error analysis in some of our top-performing models. Specifically, we manually inspected the errors of GPT-3.5-turbo (1-shot), GPT-4 (3-shot), MPNet-v2 (10-shot), and MPNet-v2 (Full-Data).

4.9.1 Errors in LLMs

The most frequently misclassified labels by GPT-4 (3-shot) and GPT-3.5-turbo (1-shot) are listed in Table 4.5 along with their error rates. Examining the samples shows that the label `Get Physical Card` was often mistaken for `Change Pin`. This may be because the word "PIN" appears in all test instances of the `Get Physical Card` class, causing their vector embeddings to fall near the decision boundary of the `Change Pin` class. For the `Transfer Not Received By Recipient` class, only 20% of samples included phrases explicitly indicating non-receipt, making classification difficult. The remaining samples involved issues with transaction timing and recipient visibility, which led them to be confused with the `Transfer Timing` class.

The third label presented in the table, `Beneficiary Not Allowed`, was incorrectly classified mostly as `Declined Transfer` by GPT-4 and GPT-3.5, respectively. The incorrect labeling could be attributed to sentences that might seem ambiguous even for a human classifier. For instance, sentences such as *"I tried to make a transfer, but it was declined."* and *"The system does not allow me to transfer funds to this beneficiary."* were misinterpreted as pertaining to `Declined Transfer` class, due to the mention of terms like *"declined"* and *"does not allow"*. The models, thus, failed to discern the actual issue of beneficiary disallowance.

Gold Label	Misclassifications	
	GPT-4	GPT-3.5
Get Physical Card	87.5%	87.5%
Transfer Not Received	62.5%	50.0%
Beneficiary Not Allowed	42.5%	60.0%

Table 4.5: Top misclassified labels, along with their misclassification percentages (out of 40 test instances), for the GPT-4 (3-shot) and GPT-3.5 models (1-shot).

4.9.2 Errors in BERT-based models

In Table 4.6, the MPNet-v2 model shows significant confusion between two labels when trained on both the full dataset and with 10 samples per class. Specifically, after inspection, we see that for the gold label `Top Up Failed`, the sentence *"please tell me why my top-up failed"* was misclassified as `Top Up Reverted`, likely due to contextual overlap between failed and reverted top-ups. Another source of confusion involved the `Transfer Timing` gold label. Sentences like *"How many days does it take until funds are in my account?"* were misclassified as `Pending Transfer` or `Balance Not Updated After Cheque or Cash Deposit`. The temporal aspect of the query seems to direct the models towards other classes that also involve time, thus highlighting challenges in differentiating specific concerns related to money transfers.

Gold Label	Misclassifications	
	Full-Data	10-shot
Top Up Failed	22.5%	17.5%
Transfer Timing	20.0%	45.0%

Table 4.6: Misclassification percentages (out of 40 test instances) for the MPNet-v2 model when trained on Full-Data vs. when trained on 10 samples per class.

4.9.3 Addressing Overlapping Labels in Future Work

We anticipate that the above error analysis (Sections 4.9.1 and 4.9.2) will inspire future work in financial intent recognition. One possible approach to reduce such errors is the use of

hierarchical classifiers (Chalkidis et al., 2020a). Such classifiers could first identify broad categories like transfers or top-ups and then further classify them into more specific labels, for example, distinguishing *pending transfer* from *transfer timing*, and *top-up failed* from *top-up reverted*. We believe these methods can be integrated into native TensorFlow or PyTorch pipelines or simulated via Chain-of-Thought prompting techniques (Wei et al., 2022), enabling LLMs to differentiate fine-grained financial labels by reasoning step-by-step or class-by-class.

4.10 Dynamic Few-Shot Prompting in other studies

While our study was one of the first ones to show the pros and cons of BERT-based models and LLMs, other researchers were investigating the same topic, but focusing on performance, whereas this study also considers cost. More specifically, there was one study (Milios et al., 2023) that got published one month later than ours (Loukas et al., 2023a), which was studying in-context learning in datasets with large label sets, including Banking77 (like we do), but including two more datasets and focusing on open-weight LLMs instead of closed-source LLMs like in our case. Just like us, Milios et al. (2023) also uses a Retrieval model to retrieve the top K most similar samples from the training set during inference, which they call "Retrieval-Augmented In-Context Learning".

The results of Milios et al. (2023) are shown in Table 4.10. Using open-weight models and testing the same method as we do in more datasets, their results validate our findings: providing the LLM with a dynamically constructed few-shot prompt can actually boost performance in intent recognition with many labels. Moreover, the results of Milios et al. (2023) show that substantial improvements over the previous state-of-the-art results on all three intent recognition datasets requires only LLaMA-2 7B, which is a relatively small language model and it can be run on consumer-grade hardware when using 8-bit quantization or less. Lastly, we should note that during the last years of the work of this thesis (2024-2025), Dynamic Few-Shot prompting got more popular with tutorials appearing about it on the LangChain and Azure blogs.[16]

[16]See `https://blog.langchain.dev/dynamic-few-shot-examples-langsmith-datasets/` and `https://techcommunity.microsoft.com/blog/fasttrackforazureblog/leveraging-dynamic-few-shot-prompt-with-azure-openai/4225235`.

Model	BANKING 77		HWU 64		CLINC 150	
	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
Pre-trained SBERT 1-NN	78.41	85.39	69.89	75.46	82.51	84.84
ConvFit (reported)	–	87.38	–	85.32	–	92.89
SetFit	79.89 ± 0.14	84.51 ± 0.60	78.38 ± 0.73	83.35 ± 0.57	88.68 ± 0.20	90.67 ± 0.29
DeBERTa	81.47 ± 1.6	88.41 ± 0.19	79.80 ± 0.81	86.93 ± 0.052	91.86 ± 0.66	95.05 ± 0.33
LLaMA 7B	84.42	87.63	85.87	87.55	88.58	91.73
LLaMA 65B	87.73	90.71	89.03	90.96	91.94	94.47
LLaMA 2 7B	86.40	89.45	87.55	87.82	94.13	95.20
LLaMA 2 7B 4K	85.91	90.58	87.17	90.33	95.36	96.02
LLaMA 2 70B	87.56	90.58	90.22	90.77	96.35	97.13
LLaMA 2 70B 4K	**88.96**	**92.11**	**90.61**	**91.73**	**97.56**	**98.18**

Table 4.7: Intent recognition accuracy for retrieval+ICL and baseline methods by Milios et al. (2023). Their "retrieval+ICL" method is essentially the same one we use and call "Dynamic Few-Shot Prompting with RAG". The retrieval/training dataset size is given by the second row of the header (10-shot is 10 examples per class, 5-shot is 5). The "4K" suffix in the LLaMA 2 models means the authors there greedily filled the context windows to explore how much gain is possible if you pack in as many examples as will fit in the model input, finding that it actually degrades performance on the 7b variant but helps the larger variant (70b).

4.11 Conclusion

We conducted a comprehensive few-shot text classification study using LLMs and BERT-based models, also discussing their trade-offs between cost and performance. To our knowledge, this research was the first to study all of those both from a performance and cost point-of-view. More specifically, we focused on Banking77, a financial intent recognition dataset with real-life challenges, such as its large number of intents and overlapping labels.[17]

The effectiveness of in-context learning with conversational LLMs is evident from our results. This approach provides a fast and practical way to achieve strong performance in few-shot scenarios, particularly in resource-constrained domains such as finance. For example, we show that LLMs like GPT-3.5-turbo, GPT-4, and Anthropic Claude 2 can outperform BERT-based models in very low-data settings (1- and 3-shot). However, fine-tuning BERT-based models such as MPNet-v2 with SetFit enables us to surpass the results of Mehri and Eric (2021) by 2.2 percentage points in the 10-shot setting. Moreover, SetFit with 5-shot

[17]This aligned with our specific use case at the time: deploying a chatbot in the financial domain for a customer with limited annotated data.

examples is competitive with LLMs in the 3-shot setting. We also observe that using expert-curated examples yields better performance than randomly selected ones. Finally, while LLM services reduce the need for technical expertise and eliminate GPU training time, their cost can be prohibitive for small organizations (e.g., $740 for 3-shot experiments on ~3k samples with GPT-4).

Following the presentation of detailed pricing costs to aid the community make better decisions, we also demonstrated a cost-effective inference method for LLMs. We utilized a semantic similarity retriever to return only a small but substantial fraction of training examples to our prompt, showing that this performs better than classic few-shot in-context learning. Most importantly, using a different model like Anthropic Claude 2 with this RAG-based approach, we can save multiple (22) times the cost associated with using GPT-4 with a standard few-shot (700$ less) while getting a higher classification score. The same findings, score-wise, were also verified by another study (Milios et al., 2023) using the same methodology, in more datasets, and focusing on open-weight LLMs instead of closed-source ones.

Lastly, to showcase how companies can utilize LLMs in scenarios with limited data, we also used GPT-4 to generate artificial data for data augmentation. We conclude that these generated data points are helpful up to a specific threshold (7 generated examples in our case), after which our performance starts to drop, possibly due to the LLM starting to generate noise.

```
You are an excellent assistant at generating high-quality data.

Your task is: Given some examples of text instances that belong to a specific label,
create k similar sentences for each class given.

Please note that these labels might contain semantic overlaps, so pay special attention to them.

Here are the examples:

text: I'm having trouble proving my identity
label: unable_to_verify_identity
text: My identity isn't being accepted, what should I do?
label: unable_to_verify_identity
text: app malfunctioning, does not know its me
label: unable_to_verify_identity
text: Which documents do I need for this identity check?
label: verify_my_identity
text: I'm not sure how to complete the identity checks process.
label: verify_my_identity
text: What are the steps for the identity checks?
label: verify_my_identity
text: Can I use my account now, even though
my identity verification has not gone through yet?
label: why_verify_identity
text: Should I complete my security verification before I can use my account?
label: why_verify_identity
text: Why does it matter that I prove my identity?
label: why_verify_identity

Now, generate your data:
```

Figure 4.6: The prompt template we used to query GPT-4 for data-augmentation in overlapping labels. We give the task and the special instruction (to pay attention to overlapping labels) to the top, then provide the data, and then remind the LLM of the task (to generate the data).

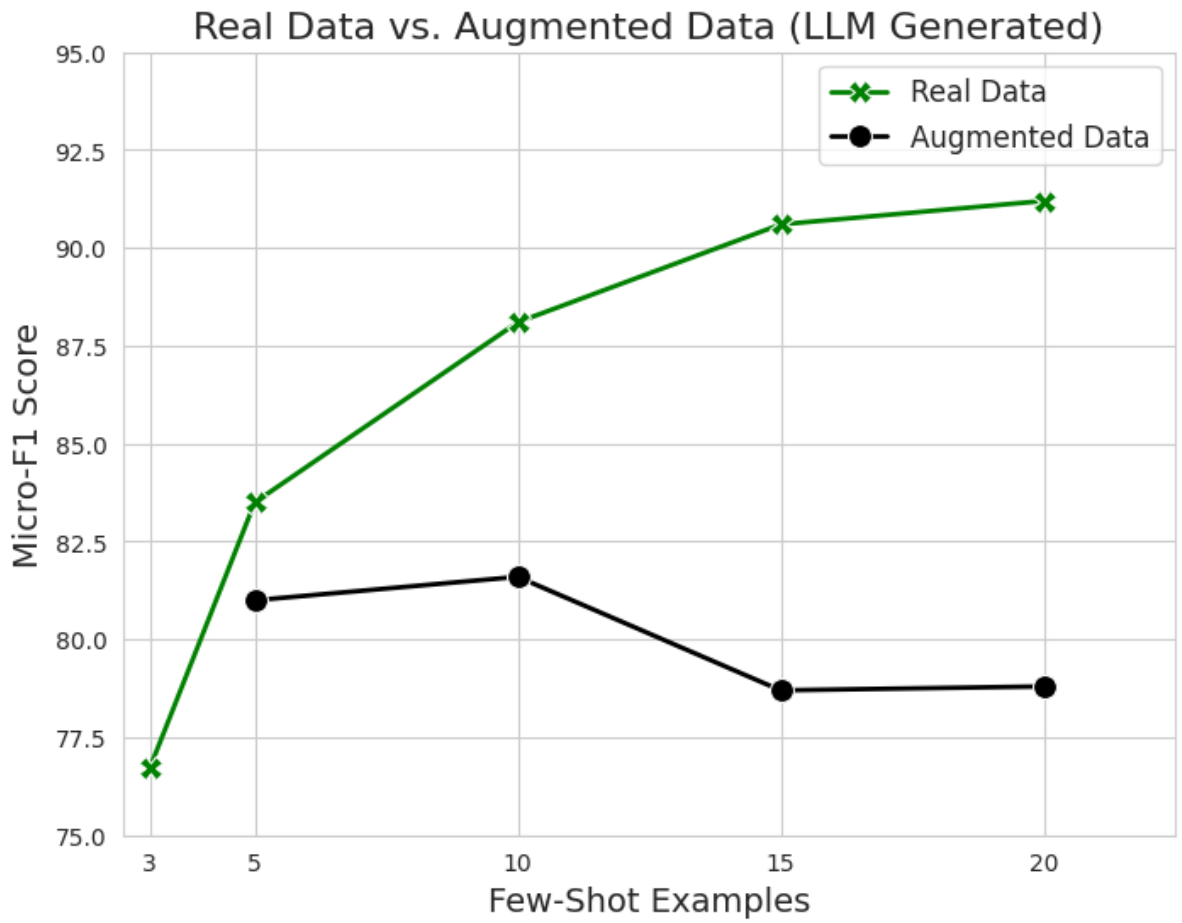


Figure 4.7: The Micro-F1 Score for various few-shot settings. In the Augmented Data (black) line, 3 of the examples each time (out of the 5, 10, 15, 20) belong to the representative samples that the human expert picked from the real data.

Chapter 5

Conclusions, Limitations and Future Work

5.1 Summary of work and conclusions

This thesis aimed to advance applied Natural Language Processing (NLP) for business and financial documents by addressing practical industry challenges across the data, application, and deployment layers, with a consistent emphasis on resource-constrained environments. We focused on answering three main research questions: (1) How can open-access, unstructured business documents be effectively leveraged for financial NLP? (2) How can current deep learning (DL) and NLP techniques be adapted to create business value in tasks like automatic document tagging, considering the nuances of financial language and its heavy reliance on numerics? (3) What are the most accurate and cost-efficient approaches for resource-limited classification, comparing BERT-based models and recent LLMs, and how can LLM deployment be optimized for cost?

This body of work has resulted in several publicly available resources to support future research and development: EDGAR-CORPUS, EDGAR-W2V, EDGAR-CRAWLER, FiNER-139, the SEC-BERT family of BERT models, and expert-curated samples for Banking77. The findings of this thesis have been peer-reviewed, presented at major ACL[1] and ACM[2] conferences and workshops, and have also resulted in a granted patent. Followingly, we provide the specific conclusions from addressing each research question.

In response to our first research question on data accessibility (Chapter 2), we addressed

[1] https://www.aclweb.org/portal/

[2] https://www.acm.org/

the challenge of data scarcity in financial NLP by focusing on the SEC EDGAR web repository. To improve access to this data, we developed and released EDGAR-CRAWLER, the first open-source Python toolkit to combine automated mining of EDGAR filings with structured parsing into a clean JSON format. At the time of the writing, EDGAR-CRAWLER has more than 420 stars in Github. Using this tool, we constructed EDGAR-CORPUS, a large-scale, publicly available corpus of parsed 10-K annual reports covering over 25 years, removing significant hurdles related to business data collection and cleaning. To demonstrate its utility, we trained EDGAR-W2V word embeddings on it and we showed that these domain-specific static embeddings outperform standard alternatives and, in some short-text use cases, even modern LLM-derived embeddings. In conclusion, these open-source resources provide a valuable foundation for financial NLP, democratizing access to key data for researchers and practitioners.

To address our second research question on adapting NLP methods for financial tasks (Chapter 3), we introduced XBRL tagging, a real-world task driven by regulatory requirements, and released FiNER-139, a large dataset of 1.1 million annotated sentences. We highlighted the task's unique challenges, including a large label set and the prevalence of numeric tokens. Our experiments revealed the significant finding that a standard BILSTM model could outperform BERT and LLMs, highlighting a key weakness in Transformer tokenization when applied to numeric-heavy domains. To solve this, we proposed a simple yet effective method of using pseudo-tokens ([NUM] and [SHAPE]) to represent numeric expressions without overfragmentation, which significantly improved BERT's performance. We also released SEC-BERT, a new family of BERT models pre-trained on financial filings that, when combined with our pseudo-tokens, achieved state-of-the-art results. Our work directly motivated subsequent research that scaled the task to thousands of labels, which remains a significant challenge even for modern LLMs like GPT-4.

Finally, to investigate our third research question on cost-efficient deployment (Chapter 4), we conducted a comprehensive few-shot text classification study on the Banking77 dataset, comparing LLMs and BERT-based models on both performance and cost. We demonstrated that while LLMs excel in 1- and 3-shot scenarios, BERT-based models like MPNet-v2 with contrastive learning (SetFit) become more effective and economical with only slightly more data (e.g., 5-10 samples per class). Recognizing the high cost of commercial LLM APIs, we demonstrated "Dynamic Few-Shot Prompting," a RAG-based method that drastically reduces inference costs multiple times while improving accuracy by retrieving only a small, relevant fraction of examples for the prompt. We also showed that using expert-curated samples in-

stead of randomly-sampled ones can boost performance, and that synthetic data from GPT-4 offers a viable, though less effective, alternative.

In conclusion, this thesis provides a practical, end-to-end contribution to applied financial NLP, moving from foundational data access to application-specific model development and cost-efficient deployment. This thesis introduces open-source tools and datasets, proposes new techniques for numeric-rich text applications, and establishes data-driven strategies for balancing performance with cost in terms of deployment. These contributions provide a valuable roadmap for researchers and practitioners, especially those working in resource-constrained environments such as small-to-medium enterprises and small research groups, to build and deploy effective and economically viable machine learning methods for financial and business language applications.

5.2 Limitations

A key limitation of the work in Chapter 2 is that the parsing logic of EDGAR-CRAWLER relies on regular expressions tailored to the structure of SEC filings. While effective, these patterns require some little maintenance as filing formats may evolve over time. However, as we demonstrated, this process can be efficiently managed through a human-in-the-loop framework augmented by AI coding assistants like Github Copilot. Similarly, the SEC-BERT models developed in Chapter 3 are pre-trained specifically on regulatory financial filings. This specialization, while powerful for regulatory document tasks, may limit their out-of-the-box performance on other financial domains with different terminologies, such as financial news or social media. Finally, the study on cost-efficient deployment in Chapter 4 was conducted on a single dataset, Banking77, primarily due to the significant costs associated with consuming commercial LLM APIs. While this is a limitation, our core findings on the effectiveness of the RAG-based approach have since been corroborated by subsequent research on multiple datasets using open-weight LLMs (Milios et al., 2023).

5.3 Future Work

Based on the findings and limitations of this thesis, several practical and interesting directions for future work emerge. First, there is the expansion of the EDGAR-CRAWLER toolkit by adding new functionalities, such as parsing documents related to insider trading activities. A key next step would also be to develop a web service for the toolkit to further democratize

access to structured financial NLP data, especially for non-technical users. Regarding the performance of language models in numeric-heavy tasks like XBRL tagging, a practical direction involves fine-tuning open-weight LLMs, potentially combining them with the numeric-aware pseudo-token strategies we developed, to create a more effective and cost-efficient solution for this extreme classification task. Lastly, there is the need for a broader analysis of cost-efficient LLM deployment strategies. This involves benchmarking the RAG-based "Dynamic Few-Shot Prompting" method against other emerging techniques, such as prompt caching (Gim et al., 2024) which is now becoming generally available in most commercial LLM APIs.

Bibliography

J. Ács, Á. Kádár, and A. Kornai. Subword pooling makes a difference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2284–2295, Online, Apr. 2021. URL https://www.aclweb.org/anthology/2021.eacl-main.194.

J. C. J. Ahn, D. Gorduza, and S. Park. Hidden neighbours: extracting industry momentum from stock networks. *Financial Markets and Portfolio Management*, 38:415–441, 2024. doi: 10.1007/s11408-024-00455-4.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, USA, June 2019. doi: 10.18653/v1/W19-1909. URL https://www.aclweb.org/anthology/W19-1909.

S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025. URL https://arxiv.org/abs/2502.13923.

A. A. Baldwin, C. E. Brown, and B. S. Trinkle. Xbrl: An impacts framework and research challenge. *Journal of Emerging Technologies in Accounting*, 3:97–116, 2006.

S. Balsam, E. Bartov, and C. Marquardt. Accruals management, investor sophistication, and

equity valuation: Evidence from 10–q filings. *Journal of Accounting Research*, 40(4): 987–1012, 2002.

L. E. Baum, T. Petrie, G. W. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970. URL https://api.semanticscholar.org/CorpusID:122568650.

I. Beltagy, K. Lo, and A. Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, Nov. 2019. doi: 10.18653/v1/D19-1371. URL https://www.aclweb.org/anthology/D19-1371.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

G. Bhatia, E. M. B. Nagoudi, H. Cavusoglu, and M. Abdul-Mageed. FinTral: A family of GPT-4 level multimodal financial large language models. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13064–13087, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.774. URL https://aclanthology.org/2024.findings-acl.774/.

D. M. Bikel, R. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*, 2016.

A. Bouzaki. Enhancing intent classification via zero-shot and few-shot chatgpt prompting engineering: Generating training data or directly detecting intents? *MSc thesis in Lan-*

guage Technology, Interdepartmental Programme, National and Kapodistrian University of Athens, 2023.

D. Braun, A. Hernandez Mendez, F. Matthes, and M. Langen. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185, Saarbrücken, Germany, Aug. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5522. URL https://aclanthology.org/W17-5522.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. teusz Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

T. Cao, N. Raman, D. Dervovic, and C. Tan. Characterizing multimodal long-form summarization: A case study on financial reports, 2024. URL https://arxiv.org/abs/2404.06162.

M. E. Carter and B. S. Soo. The relevance of form 8-k reports. *Journal of Accounting Research*, 37(1):119–132, 1999.

I. Casanueva, T. Temčinas, D. Gerz, M. Henderson, and I. Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5.

R. A. Cazier and R. J. Pfeiffer. Why are 10-k filings so long? *Accounting Horizons*, 30(1): 1–21, Mar 2016. ISSN 0888-7993. doi: 10.2308/acch-51240. URL https://doi.org/10.2308/acch-51240.

D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural*

Language Processing: System Demonstrations, pages 169–174, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2029. URL https://aclanthology.org/D18-2029.

I. Chalkidis, M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos. Neural contract element extraction revisited. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019. URL https://openreview.net/forum?id=B1x6fa95UH.

I. Chalkidis, M. Fergadiotis, S. Kotitsas, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. An empirical study on large-scale multi-label text classification including few and zero-shot labels. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7503–7515, Online, Nov. 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.607. URL https://aclanthology.org/2020.emnlp-main.607.

I. Chalkidis, M. Fergadiotis, S. Kotitsas, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. An empirical study on large-scale multi-label text classification including few and zero-shot labels. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 7503–7515, 2020b. doi: 10.18653/v1/2020.emnlp-main.607. URL https://doi.org/10.18653/v1/2020.emnlp-main.607.

I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online, Nov. 2020c. doi: 10.18653/v1/2020.findings-emnlp.261. URL https://www.aclweb.org/anthology/2020.findings-emnlp.261.

C.-C. Chen, H.-H. Huang, H. Takamura, and H.-H. Chen, editors. *Proceedings of the Fourth Workshop on Financial Technology and Natural Language Processing (FinNLP)*, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.finnlp-1.0.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL

https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *ArXiv*, abs/1805.10190, 2018.

L. Crane, E. Green, M. Harnish, W. McClennan, P. Soto, B. Vrankovich, and J. Williams. Tracking real time layoffs with sec filings: A preliminary investigation. *Finance and Economics Discussion Series*, pages 1–30, 04 2024. doi: 10.17016/FEDS.2024.020.

T. Daudert and S. Ahmadi. CoFiF: A corpus of financial reports in French language. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, pages 21–26, Macao, China, 12 Aug. 2019. URL https://www.aclweb.org/anthology/W19-5504.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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*, pages 4171–4186, Minneapolis, Minnesota, June 2019a. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *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, Minneapolis, Minnesota, June 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf.

J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020.

T. Dyer, M. Lang, and L. Stice-Lawrence. The evolution of 10-k textual disclosure: Evidence from latent dirichlet allocation. *Journal of Accounting and Economics*, 64(2):221–245, 2017. doi: 10.1016/j.jacceco.2017.07.002.

M. El-Haj, P. Rayson, M. Walker, S. Young, and V. Simaki. In search of meaning: Lessons, resources and next steps for computational analysis of financial discourse. *Research Methods & Methodology in Accounting eJournal*, 2019. URL https://api.semanticscholar.org/CorpusID:88523877.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June 2005. doi: 10.3115/1219840.1219885. URL https://www.aclweb.org/anthology/P05-1045.

I. E. Fisher, M. R. Garnsey, and M. E. Hughes. Natural language processing in accounting, auditing and finance: A synthesis of the literature with a roadmap for future research. *Intelligent Systems in Accounting, Finance and Management*, 23(3):157–214, 2016. doi: 10.1002/isaf.1386.

S. Francis, J. Van Landeghem, and M.-F. Moens. Transfer learning for named entity recognition in financial and biomedical documents. *Information*, 10(8):248, 2019.

P. Gandhi, T. Loughran, and B. McDonald. Using Annual Report Sentiment as a Proxy for Financial Distress in U.S. Banks. *Journal of Behavioral Finance*, 20(4):424–436, October 2019. doi: 10.1080/15427560.2019.155. URL https://ideas.repec.org/a/taf/hbhfxx/v20y2019i4p424-436.html.

G. Gauthier-melancon, O. Marquez Ayala, L. Brin, C. Tyler, F. Branchaud-charron, J. Marinier, K. Grande, and D. Le. Azimuth: Systematic error analysis for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 298–310, Abu Dhabi, UAE, Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-demos.30.

M. Geva, A. Gupta, and J. Berant. Injecting numerical reasoning skills into language models. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 946–958, 2020. doi: 10.18653/v1/2020.acl-main.89. URL https://doi.org/10.18653/v1/2020.acl-main.89.

I. Gim, G. Chen, S. seob Lee, N. Sarda, A. Khandelwal, and L. Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/a66caa1703fe34705a4368c3014c1966-Abstract-Conference.html.

N. C. Gkoumas, G. N. Leledakis, E. G. Pyrgiotakis, and I. Androutsopoulos. Bank competition, loan portfolio concentration and stock price crash risk: The role of tone ambiguity. *British Journal of Management*, 2024. URL https://api.semanticscholar.org/CorpusID:270945609.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Italy, 13–15 May 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.

S. Goel and J. Gangolly. Beyond the numbers: Mining the annual reports for hidden cues indicative of financial statement fraud. *Intelligent Systems in Accounting, Finance and Management*, 19(2):75–89, 2012. doi: 10.1002/isaf.1326. URL https://doi.org/10.1002/isaf.1326.

S. Goel and Ö. Uzuner. Do sentiments matter in fraud detection? estimating semantic orientation of annual reports. *Intelligent Systems in Accounting, Finance and Management*, 23: 215–239, 2016.

E. Goldman and J. Goldberger. Crf with deep class embedding for large scale classification. *Computer Vision and Image Understanding*, 191:102865, 2020. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2019.102865. URL https://www.sciencedirect.com/science/article/pii/S1077314219301560.

A. Graves, N. Jaitly, and A. Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Olomouc, Czech Republic, 2013.

P. A. Griffin. Got information? investor response to form 10-k and form 10-q edgar filings. *Review of Accounting Studies*, 8:433–460, 2003.

M. Günther, S. Sturua, M. K. Akram, I. Mohr, A. Ungureanu, B. Wang, S. Eslami, S. Martens, M. Werk, N. Wang, and H. Xiao. jina-embeddings-v4: Universal Embeddings for Multimodal Multilingual Retrieval. *arXiv preprint arXiv:2506.18902*, 2025. URL https://arxiv.org/abs/2506.18902.

L. Guo, F. Shi, and J. Tu. Textual analysis and machine learning: Crack unstructured data in finance and accounting. *Journal of Finance and Data Science*, 2(3):153–170, 2016. doi: 10.1016/j.jfds.2017.02.001.

U. Hahn, V. Hoste, and M.-F. Tsai, editors. *Proceedings of the First Workshop on Economics and Natural Language Processing*, Melbourne, Australia, July 2018. URL https://aclanthology.org/W18-3100.

P. Hampton, H. Wang, W. Blackburn, and Z. Lin. Automated sequence tagging: Applications in financial hybrid systems. In *Research and Development in Intelligent Systems XXXIII*, pages 295–306. Springer, 2016. doi: 10.1007/978-3-319-47175-4_22. URL https://doi.org/10.1007/978-3-319-47175-4_22.

P. J. Hampton, H. Wang, and W. Blackburn. A hybrid ensemble for classifying and repurposing financial entities. In *Research and Development in Intelligent Systems XXXII*, pages 197–202. Springer, 2015. doi: 10.1007/978-3-319-25032-8_15. URL https://doi.org/10.1007/978-3-319-25032-8_15.

S. G. Händschke, S. Buechel, J. Goldenstein, P. Poschmann, T. Duan, P. Walgenbach, and U. Hahn. A corpus of corporate annual and social responsibility reports: 280 million tokens of balanced organizational writing. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 20–31, Melbourne, Australia, July 2018. doi: 10.18653/v1/W18-3103. URL https://aclanthology.org/W18-3103.

B. He, L. I. Mostrom, and A. Sufi. Investing in customer capital. (33171), November 2024. doi: 10.3386/w33171. URL http://www.nber.org/papers/w33171.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

L. P. Hillebrand, T. Deußer, T. D. Khameneh, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa. Kpi-bert: A joint named entity recognition and relation extraction model for financial reports. *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 606–612, 2022. URL https://api.semanticscholar.org/CorpusID:251280287.

R. Hoitash and U. Hoitash. Measuring accounting reporting complexity with xbrl. *The Accounting Review*, 93:259–287, 2018.

M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015. URL http://arxiv.org/abs/1508.01991.

S. B. Juyeon Kang, Ismail El Maarouf and M. Gan. Finsim-3: The 3rd shared task on learning semantic similarities for the financial domain. In *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, Held Online, Aug. 2021.

A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J.-B. Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehri, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wi-

eting, J. Lai, J. Orbay, J. Fernandez, J. Newlan, J. yeong Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson, M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Põder, S. Bhatnagar, S. R. Panyam, S. Eiger, S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evci, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed, V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, D. D. Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, L. Hussenot, and . additional authors. Gemma 3 technical report. *arXiv*, 2025. doi: 10.48550/arXiv.2503.19786. URL `https://doi.org/10.48550/arXiv.2503.19786`. arXiv:2503.19786.

A. G. Katsafados, G. N. Leledakis, E. G. Pyrgiotakis, I. Androutsopoulos, I. Chalkidis, and M. Fergadiotis. Textual information and ipo underpricing: A machine learning approach. *The Journal of Financial Data Science*, 5(2):100–135, Spring 2023a. doi: 10.3905/jfds.2023.1.121.

A. G. Katsafados, G. N. Leledakis, E. G. Pyrgiotakis, I. Androutsopoulos, and M. Fergadiotis. Machine learning in bank merger prediction: A text-based approach. *European Journal of Operational Research*, 312:783–797, 2023b. URL `https://api.semanticscholar.org/CorpusID:260647672`.

C. Kearney and S. Liu. Textual sentiment in finance: A survey of methods and models. *International Review of Financial Analysis*, 33:171–185, 2014. doi: 10.1016/j.irfa.2014.02.006.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus–a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–i182, 2003.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015,*

San Diego, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280, Boulder, Colorado, June 2009. URL https://aclanthology.org/N09-1031.

A. Kumar, H. Alam, T. Werner, and M. Vyas. Experiments in candidate phrase selection for financial named entity extraction - a demo. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 45–48, Osaka, Japan, Dec. 2016. URL https://www.aclweb.org/anthology/C16-2010.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, USA, 2001. ISBN 1558607781.

M. Lamm, A. Chaganty, C. D. Manning, D. Jurafsky, and P. S. Liang. Textual analogy parsing: What's shared and what's compared among analogous facts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1776–1788, 2018.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 260–270, San Diego, California, 2016a.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016b. doi: 10.18653/v1/N16-1030. URL https://www.aclweb.org/anthology/N16-1030.

S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang, and J. Mars. An evaluation dataset for intent classification

and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1131. URL https://aclanthology.org/D19-1131.

H. Lee, M. Surdeanu, B. MacCartney, and D. Jurafsky. On the importance of text analysis for stock price prediction. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1170–1175, Reykjavik, Iceland, May 2014.

Y.-J. Lee. The effect of quarterly report readability on information efficiency of stock prices. *Contemporary Accounting Research*, 29(4), 2012.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

X. Li, W. Aitken, X. Zhu, and S. W. Thomas. Learning better intent representations for financial open intent classification. In *Proceedings of the Fourth Workshop on Financial Technology and Natural Language Processing (FinNLP)*, pages 68–77, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.finnlp-1.8.

B. Y. Lin, F. Xu, Z. Luo, and K. Zhu. Multi-channel BiLSTM-CRF model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 160–165, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4421. URL https://aclanthology.org/W17-4421.

J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10.

N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts, 2023.

X. Liu, A. Eshghi, P. Swietojanski, and V. Rieser. *Benchmarking Natural Language Understanding Services for Building Conversational Agents*, pages 165–183. Springer Singapore, Singapore, 2021. ISBN 978-981-15-9323-9. doi: 10.1007/978-981-15-9323-9_15. URL https://doi.org/10.1007/978-981-15-9323-9_15.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

T. Loughran and B. McDonald. Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230, 2016. doi: 10.1111/1475-679X.12123.

T. Loughran and B. McDonald. Textual analysis in finance. *Annual Review of Financial Economics*, 12(1):357–375, 2020. doi: 10.1146/annurev-financial-012820-103159.

L. Loukas, K. Bougiatiotis, M. Fergadiotis, D. Mavroeidis, and E. Zavitsanos. DICE at FinSim-3: Financial Hypernym Detection using Augmented Terms and Distance-based Features. In *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, pages 40–45, Online, 19 Aug. 2021a. -. URL https://aclanthology.org/2021.finnlp-1.7.

L. Loukas, M. Fergadiotis, I. Androutsopoulos, and P. Malakasiotis. EDGAR-CORPUS: Billions of tokens make the world go round. In U. Hahn, V. Hoste, and A. Stent, editors, *Proceedings of the Third Workshop on Economics and Natural Language Processing*, pages 13–18, Punta Cana, Dominican Republic, Nov. 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.econlp-1.2. URL https://aclanthology.org/2021.econlp-1.2/.

L. Loukas, M. Fergadiotis, I. Androutsopoulos, and P. Malakasiotis. EDGAR-CORPUS: Billions of Tokens Make The World Go Round. In *Proceedings of the Third Workshop on Economics and Natural Language Processing*, pages 13–18, Punta Cana, Dominican Republic, Nov. 2021c. Association for Computational Linguistics. doi: 10.18653/v1/2021.econlp-1.2. URL https://aclanthology.org/2021.econlp-1.2.

L. Loukas, E. Spyropoulou, P. Malakasiotis, M. Fergadiotis, I. Chalkidis, I. Androutsopoulos, and G. Paliouras. System and method for automatically tagging documents. In *European Office (EPO) Patent Application 21 386 048.9*, 2021d.

L. Loukas, E. Spyropoulou, P. Malakasiotis, M. Fergadiotis, I. Chalkidis, I. Androutsopoulos, and G. Paliouras. System and method for automatically tagging documents. In *United States Patent and Trademark Office (USPTO) Patent US17/873,932*, 2021e.

L. Loukas, E. Spyropoulou, P. Malakasiotis, M. Fergadiotis, I. Chalkidis, I. Androutsopoulos, and G. Paliouras. System and method for automatically tagging documents. In *World Intellectual Property Organization (WIPO) Patent Application WO2023006773A1*, 2021f.

L. Loukas, M. Fergadiotis, I. Chalkidis, E. Spyropoulou, P. Malakasiotis, I. Androutsopoulos, and G. Paliouras. FiNER: Financial Numeric Entity Recognition for XBRL Tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4419–4431, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.303. URL https://aclanthology.org/2022.acl-long.303.

L. Loukas, I. Stogiannidis, O. Diamantopoulos, P. Malakasiotis, and S. Vassos. Making LLMs Worth Every Penny: Resource-Limited Text Classification in Banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 392–400, New York, NY, USA, 2023a. Association for Computing Machinery. ISBN 9798400702402. doi: 10.1145/3604237.3626891. URL https://doi.org/10.1145/3604237.3626891.

L. Loukas, I. Stogiannidis, P. Malakasiotis, and S. Vassos. Breaking the Bank with ChatGPT: Few-Shot Text Classification for Finance. In *Proceedings of the Fifth Workshop on Financial Technology and Natural Language Processing and the Second Multimodal AI For Financial Forecasting*, pages 74–80, Macao, 20 Aug. 2023b. URL https://aclanthology.org/2023.finnlp-1.7.

L. Loukas, F. Billert, M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos. EDGAR-CRAWLER: From Raw Web Documents to Structured Financial NLP Datasets. In *Companion Proceedings of the 32nd ACM Web Conference 2025 (WWW 2025)*, 2025a. https://github.com/lefterisloukas/edgar-crawler.

L. Loukas, N. Smyrnioudis, C. Dikonomaki, S. Barbakos, A. Toumazatos, J. Koutsikakis, M. Kyriakakis, M. Georgiou, S. Vassos, J. Pavlopoulos, and I. Androutsopoulos. GR-NLP-TOOLKIT: An Open-Source NLP Toolkit for Modern Greek. In *Proceedings of the 31st International Conference on Computational Linguistics: System Demonstrations*, pages 174–182, Abu Dhabi, UAE, Jan. 2025b. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-demos.17/.

D. Lu, H. Wu, J. Liang, Y. Xu, Q. He, Y. Geng, M. Han, Y. Xin, and Y. Xiao. Bbt-fin: Comprehensive construction of chinese financial domain pre-trained language model, corpus and benchmark, 2023. URL https://arxiv.org/abs/2302.09432.

Y.-T. Lu and Y. Huo. Financial named entity recognition: How far can LLM go? In C.-C. Chen, A. Moreno-Sandoval, J. Huang, Q. Xie, S. Ananiadou, and H.-H. Chen, editors, *Proceedings of the Joint Workshop of the 9th Financial Technology and Natural Language Processing (FinNLP), the 6th Financial Narrative Processing (FNP), and the 1st Workshop on Large Language Models for Finance and Legal (LLMFinLegal)*, pages 164–168, Abu Dhabi, UAE, Jan. 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.finnlp-1.15/.

X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, Aug. 2016. doi: 10.18653/v1/P16-1101. URL https://www.aclweb.org/anthology/P16-1101.

M. Maia, S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, and A. Balahur. Www'18 open challenge: Financial opinion mining and question answering. *Companion Proceedings of the The Web Conference 2018*, 2018.

N. Manginas, I. Chalkidis, and P. Malakasiotis. Layer-wise guided training for BERT: Learning incrementally refined document representations. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 53–61, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.spnlp-1.7. URL https://aclanthology.org/2020.spnlp-1.7.

L. McInnes, J. Healy, N. Saul, and L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

S. Mehri and M. Eric. Example-driven intent prediction with observers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2979–2992, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.237. URL https://aclanthology.org/2021.naacl-main.237.

D. Mehta and G. Wang. ICAIF '24: Proceedings of the 5th ACM International Conference on AI in Finance. 2024.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013b. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

A. Milios, S. Reddy, and D. Bahdanau. In-context learning for text classification with many labels. In D. Hupkes, V. Dankers, K. Batsuren, K. Sinha, A. Kazemnejad, C. Christodoulopoulos, R. Cotterell, and E. Bruni, editors, *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 173–184, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.genbench-1.14. URL https://aclanthology.org/2023.genbench-1.14/.

R. Moriarty, H. Ly, E. Lan, and S. McIntosh. Deal or no deal: Predicting mergers and acquisitions at scale. *2019 IEEE International Conference on Big Data (Big Data)*, pages 5552–5558, 2019.

N. Muennighoff, L. Tunstall, T. Van den Eynde, J. Müller, M. Radev, B. Ponweiser, N. Reimers, B. Wang, B. Minixhofer, J. Vayrynen, C. Zheng, J. Gusak, M. S. Hamalainen, Q. Lhoest, I. Gurevych, and T. Wolf. Mteb: Massive text embedding benchmark. In *Pro-*

ceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023), 2023. URL https://aclanthology.org/2023.emnlp-main.1182/.

A. K. Nassirtoussi, S. R. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo. Text mining for market prediction: A systematic review. *Expert Syst. Appl.*, 41:7653–7670, 2014. URL https://api.semanticscholar.org/CorpusID:10869039.

C. G. Northcutt, L. Jiang, and I. L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR)*, 70:1373–1411, 2021.

OpenAI. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

OpenAI. Gpt-4 technical report, 2023.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014a. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014b.

S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July 2012. URL https://www.aclweb.org/anthology/W12-4501.

L. Purda and D. Skillicorn. Accounting variables, deception, and a bag of words: Assessing the tools of fraud detection. *Contemporary Accounting Research*, 32(3):1193–1223, 2015.

doi: https://doi.org/10.1111/1911-3846.12089. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1911-3846.12089.

A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Y. Razeghi, R. L. Logan IV, M. Gardner, and S. Singh. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.59.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. http://is.muni.cz/publication/884893/en.

L. Reynolds and K. McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380959. doi: 10.1145/3411763.3451760. URL https://doi.org/10.1145/3411763.3451760.

G. Sahu, P. Rodriguez, I. Laradji, P. Atighehchian, D. Vazquez, and D. Bahdanau. Data augmentation for intent classification with off-the-shelf large language models. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 47–57, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.nlp4convai-1.5. URL https://aclanthology.org/2022.nlp4convai-1.5.

J. C. Salinas Alvarado, K. Verspoor, and T. Baldwin. Domain adaption of named entity recognition to support credit risk assessment. In *Proceedings of the Australasian Language Technology Association Workshop 2015*, pages 84–90, Parramatta, Australia, Dec. 2015. URL https://www.aclweb.org/anthology/U15-1010.

T. Schick and H. Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20/.

J. Serrà and A. Karatzoglou. Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, page 279–287, New York, USA, 2017. ISBN 9781450346528. doi: 10.1145/3109859.3109876. URL https://doi.org/10.1145/3109859.3109876.

A. Shah, A. Gullapalli, R. Vithani, M. Galarnyk, and S. Chava. Finer-ord: Financial named entity recognition open research dataset. *arXiv preprint arXiv:2302.11157*, 2024.

S. Sharma, S. Khatuya, M. Hegde, A. Shaikh, K. Dasgupta, P. Goyal, and N. Ganguly. Financial numeric extreme labelling: A dataset and benchmarking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3550–3561, Toronto, Canada, July 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-acl.219.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, 2012. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

I. Stogiannidis, S. Vassos, P. Malakasiotis, and I. Androutsopoulos. Cache me if you can: an online cost-aware teacher-student framework to reduce the calls to large language models. In *Findings of the Conference on Empirical Methods in Natural Language Processing*, Singapore, December 2023. Association for Computational Linguistics.

M. T. T. Teichmann and R. Cipolla. Convolutional crfs for semantic segmentation. *CoRR*, abs/1805.04777, 2018. URL http://arxiv.org/abs/1805.04777.

A. Thawani, J. Pujara, F. Ilievski, and P. Szekely. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, Online, June 2021. doi: 10.18653/v1/2021.naacl-main.53. URL https://aclanthology.org/2021.naacl-main.53.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

M.-F. Tsai, C.-J. Wang, and P.-C. Chien. Discovering finance keywords via continuous-space language models. *ACM Transactions on Management Information Systems*, 7(3), Aug. 2016. ISSN 2158-656X. doi: 10.1145/2948072. URL https://doi.org/10.1145/2948072.

L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg. Efficient few-shot learning without prompts. 2022. URL https://neurips2022-enlsp.github.io/papers/paper_17.pdf.

S. Vassos, S. Goudelis, D. Balaouras, G. Vitalis, V. Nakos, G. Pigka, L. Tsagkli, Z. Tsionas, A. Chasanis, S. van de Burgt, M. Pors, S. Papadoudis, and L. Loukas. Now I know! Empowering Voters with RAG-enabled LLMs to Eliminate Political Uncertainty. In *Proceedings of the 13th Hellenic Conference on Artificial Intelligence*, SETN '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400709821. doi: 10.1145/3688671.3688784. URL https://doi.org/10.1145/3688671.3688784.

A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. pages 5998–6008, 2017.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13:260–269, 1967. URL https://api.semanticscholar.org/CorpusID:15843983.

C. Wallek and T. Ebert. The annual report algorithm: Retrieval of financial statements. In *Proceedings of Computer Science & Information Technology)*, pages 162–176, 2016. URL https://scispace.com/pdf/the-annual-report-algorithm-retrieval-of-financial-1fg8yoqdr1.pdf.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2022.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann. Bloomberggpt: A large language model for finance, 2023. URL https://arxiv.org/abs/2303.17564.

Q. Xie, W. Han, Z. Chen, R. Xiang, X. Zhang, Y. He, M. Xiao, D. Li, Y. Dai, D. Feng, Y. Xu, H. Kang, Z. Kuang, C. Yuan, K. Yang, Z. Luo, T. Zhang, Z. Liu, G. Xiong, Z. Deng, Y. Jiang, Z. Yao, H. Li, Y. Yu, G. Hu, J. Huang, X.-Y. Liu, A. Lopez-Lira, B. Wang, Y. Lai, H. Wang, M. Peng, S. Ananiadou, and J. Huang. Finben: A holistic financial benchmark for large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 95716–95743. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/adb1d9fa8be4576d28703b396b82ba1b-Paper-Datasets_and_Benchmarks_Track.pdf.

F. Xing, E. Cambria, and R. E. Welsch. Natural language based financial forecasting: a survey. *Artificial Intelligence Review*, 50:49 – 73, 2017. URL https://api.semanticscholar.org/CorpusID:207079655.

Y. Yang, M. C. S. UY, and A. Huang. Finbert: A pretrained language model for financial communications. *arXiv*, abs/2006.08097, 2020.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Curran Associates Inc., Red Hook, NY, USA, 2019.

C. Ying and S. Thomas. Label errors in BANKING77. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 139–143, Dublin, Ireland, 2022.

Association for Computational Linguistics. doi: 10.18653/v1/2022.insights-1.19. URL https://aclanthology.org/2022.insights-1.19.

K. M. Yoo, D. Park, J. Kang, S.-W. Lee, and W. Park. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.192. URL https://aclanthology.org/2021.findings-emnlp.192.

R. You, S. Dai, Z. Zhang, H. Mamitsuka, and S. Zhu. AttentionXML: Extreme Multi-Label Text Classification with Multi-Label Attention Based Recurrent Neural Networks. *CoRR*, abs/1811.01727, 2018.

R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *Advances in Neural Information Processing Systems 32*, pages 5812–5822. Curran Associates, Inc., 2019.

E. Zavitsanos, D. Mavroeidis, K. Bougiatiotis, E. Spyropoulou, L. Loukas, and G. Paliouras. Financial misstatement detection: A realistic evaluation. In *Proceedings of the Second ACM International Conference on AI in Finance*, ICAIF '21, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391481. doi: 10.1145/3490354.3494453. URL https://doi.org/10.1145/3490354.3494453.

H. Zawbaa, W. Rashwan, S. Dutta, and H. Assem. Improved out-of-scope intent classification with dual encoding and threshold-based re-classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8708–8718, Torino, Italia, 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.763/.

X. Zhang, D. Ramachandran, I. Tenney, Y. Elazar, and D. Roth. Do language embeddings capture scales? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4889–4896, Online, Nov. 2020. doi: 10.18653/v1/2020.findings-emnlp.439. URL https://aclanthology.org/2020.findings-emnlp.439.

Appendix A

In this Appendix, we include the most relevant parts of the item extraction module of EDGAR-CRAWLER (extract_items.py), which parses the unstructured SEC filings into structured JSON output. This module is responsible for the matching mechanism powered by regular expressions, as explained in Chapter 2. The complete codebase can be found on Github.[1]

```
import re
from typing import Any, Dict, List, Optional, Tuple

# Roman numeral map needed needed for 10-Q reports.
roman_numeral_map = {
    "1": "I",
    "2": "II",
    "3": "III",
    "4": "IV",
    "5": "V",
    "6": "VI",
    "7": "VII",
    "8": "VIII",
    "9": "IX",
    "10": "X",
    "11": "XI",
    "12": "XII",
    "13": "XIII",
    "14": "XIV",
    "15": "XV",
    "16": "XVI",
    "17": "XVII",
    "18": "XVIII",
    "19": "XIX",
    "20": "XX",
}

regex_flags = re.IGNORECASE | re.DOTALL | re.MULTILINE


def adjust_item_patterns(item_index: str) -> str:
    """
    Adjust the item_pattern for matching in the document text depending on the item index. This is necessary on a case
         by case basis.

    Args:
        item_index (str): The item index to adjust the pattern for.
                          For 10-Q preprocessing, this can also be part_1 or part_2.

    Returns:
        item_index_pattern (str): The adjusted item pattern
    """

    # For 10-Q reports, we have two parts of items: part1 and part2
    if "part" in item_index:
        if "__" not in item_index:
            # We are searching for the general part, not a specific item (e.g. PART I)
            item_index_number = item_index.split("_")[1]
            item_index_pattern = rf"PART\s*(?:{roman_numeral_map[item_index_number]}|{item_index_number})"
            return item_index_pattern
        else:
            # We are working with an item, but we just consider the string after the part as the item_index
            item_index = item_index.split("__")[1]

    # Create a regex pattern from the item index
    item_index_pattern = item_index
```

[1] https://github.com/lefterisloukas/edgar-crawler/

```
    # Modify the item index format for matching in the text
    if item_index == "9A":
        item_index_pattern = item_index_pattern.replace(
            "A", r"[^\S\r\n]*A(?:\(T\))?"
        )  # Regex pattern for item index "9A"
    elif item_index == "SIGNATURE":
        # Quit here so the A in SIGNATURE is not changed
        pass
    elif "A" in item_index:
        item_index_pattern = item_index_pattern.replace(
            "A", r"[^\S\r\n]*A"
        )  # Regex pattern for other "A" item indexes
    elif "B" in item_index:
        item_index_pattern = item_index_pattern.replace(
            "B", r"[^\S\r\n]*B"
        )  # Regex pattern for "B" item indexes
    elif "C" in item_index:
        item_index_pattern = item_index_pattern.replace(
            "C", r"[^\S\r\n]*C"
        )  # Regex pattern for "C" item indexes

    # If the item is SIGNATURE, we don't want to look for ITEM
    if item_index == "SIGNATURE":
        # Some reports have SIGNATURES or Signature(s) instead of SIGNATURE
        item_index_pattern = rf"{item_index}(s|\(s\))?"
    else:
        if "." in item_index:
            # We need to escape the '.', otherwise it will be treated as a special character - for 8Ks
            item_index = item_index.replace(".", r"\.")
        if item_index in roman_numeral_map:
            # Rarely, reports use roman numerals for the item indexes. For 8-K, we assume this does not occur (due to
                their format - e.g. 5.01)
            item_index = f"(?:{roman_numeral_map[item_index]}|{item_index})"
        item_index_pattern = rf"ITEMS?\s*{item_index}"

    return item_index_pattern


def parse_item(
    text: str,
    item_index: str,
    next_item_list: List[str],
    positions: List[int],
    items_list: List[str],
    ignore_matches: int = 0,
) -> Tuple[str, List[int]]:
    """
    Parses the specified item/section in a report text.

    Args:
        text (str): The report text.
        item_index (str): Number of the requested Item/Section of the report text.
        next_item_list (List[str]): List of possible next report item sections.
        positions (List[int]): List of the end positions of previous item sections.
        ignore_matches (int): Default is 0. If positive, we skip the first [value] matches. Only used for 10-Q part
            extraction.

    Returns:
        Tuple[str, List[int]]: The item/section as a text string and the updated end positions of item sections.
    """

    # Set the regex flags
    regex_flags = re.IGNORECASE | re.DOTALL

    # Adjust the item index pattern
    item_index_pattern = adjust_item_patterns(item_index)

    # Determine the current part in case of 10-Q reports
    if "part" in item_index and "PART" not in item_index_pattern:
        item_index_part_number = item_index.split("__")[0]

    possible_sections_list = []  # possible list of (start, end) matches
    impossible_match = None  # list of matches where no possible section was found - (start, None) matches
    last_item = True
    for next_item_index in next_item_list:
        # Check if the next item is the last one
        last_item = False
        if possible_sections_list:
            break
        if next_item_index == next_item_list[-1]:
            last_item = True

        # Adjust the next item index pattern
        next_item_index_pattern = adjust_item_patterns(next_item_index)

        # Check if the next item is in a different part - in this case we exit the loop
        if "part" in next_item_index and "PART" not in next_item_index_pattern:
            next_item_index_part_number = next_item_index.split("__")[0]
            if next_item_index_part_number != item_index_part_number:
                # If the next item is in a subsequent part, we won't find it in the text -> should simply extract the
                    rest of the current part
                last_item = True
                break
```

```
        # Find all the text sections between the current item and the next item
        matches = list(
            re.finditer(
                rf"\n[^\S\r\n]*{item_index_pattern}[.*~\-:\s\(]",
                text,
                flags=regex_flags,
            )
        )
        for i, match in enumerate(matches):
            if i < ignore_matches:
                # In some cases, the first matches might capture longer sections because parts/items are mentioned in
                    the ToC.
                # We detect this in another place and then skip the first [ignore_matches] matches until we are more
                    certain to have the correct section.
                continue
            offset = match.start()

            # First we do a case-sensitive search. This is because in some reports, parts or items are mentioned in the
                 content,
            # which we don't want to detect as a section header.
            # The section headers are usually in uppercase, so checking this first avoids some errors.
            possible = list(
                re.finditer(
                    rf"\n[^\S\r\n]*{item_index_pattern}[.*~\-:\s\()]"
                    ".+?"
                    rf"(\n[^\S\r\n]*{str(next_item_index_pattern)}[.*~\-:\s\(])",
                    text[offset:],
                    flags=re.DOTALL,
                )
            )

            if not possible:
                # If there is no match, follow with a case-insensitive search
                possible = list(
                    re.finditer(
                        rf"\n[^\S\r\n]*{item_index_pattern}[.*~\-:\s\()]"
                        ".+?"
                        rf"(\n[^\S\r\n]*{str(next_item_index_pattern)}[.*~\-:\s\(])",
                        text[offset:],
                        flags=regex_flags,
                    )
                )

            # If there is a match, add it to the list of possible sections
            if possible:
                possible_sections_list += [(offset, possible)]
            elif (
                next_item_index == next_item_list[-1]
                and not possible_sections_list
                and match
            ):
                # If there is no (start, end) section, there might only be a single item in the report (can happen for
                    8-K)
                impossible_match = match

    # Extract the wanted section from the text
    item_section, positions = get_item_section(possible_sections_list, text, positions)

    # If item is the last one (usual case when dealing with EDGAR's old .txt files), get all the text from its
         beginning until EOF.
    if positions:
        # If the item is the last one, get all the text from its beginning until EOF
        # This is needed in cases where the SIGNATURE section cannot be found
        if item_index in items_list and item_section == "":
            item_section = get_last_item_section(item_index, text, positions)
        # SIGNATURE is the last one, get all the text from its beginning until EOF
        if item_index == "SIGNATURE":
            item_section = get_last_item_section(item_index, text, positions)
    elif impossible_match or last_item:
        # If there is only a single item in a report and no SIGNATURE (can happen for 8-K reports),
        # 'possible_sections_list' and thus also 'positions' will always be empty.
        # In this case we just want to extract from the match until the end of the document
        if item_index in items_list:
            item_section = get_last_item_section(item_index, text, positions)

    return item_section, positions


def get_item_section(
    possible_sections_list: List[Tuple[int, List[re.Match]]],
    text: str,
    positions: List[int],
) -> Tuple[str, List[int]]:
    """
    Returns the correct section from a list of all possible item sections.

    Args:
        possible_sections_list: List containing all the possible sections between Item X and Item Y.
        text: The whole text.
        positions: List of the end positions of previous item sections.

    Returns:
```

```
        Tuple[str, List[int]]: The correct section and the updated list of end positions.
    """

    # Initialize variables
    item_section: str = ""
    max_match_length: int = 0
    max_match: Optional[re.Match] = None
    max_match_offset: Optional[int] = None

    # Find the match with the largest section
    for offset, matches in possible_sections_list:
        # Find the match with the largest section
        for match in matches:
            match_length = match.end() - match.start()
            # If there are previous item sections, check if the current match is after the last item section
            if positions:
                if (
                    match_length > max_match_length
                    and offset + match.start() >= positions[-1]
                ):
                    max_match = match
                    max_match_offset = offset
                    max_match_length = match_length
            # If there are no previous item sections, just get the first match
            elif match_length > max_match_length:
                max_match = match
                max_match_offset = offset
                max_match_length = match_length

    # Return the text section inside that match
    if max_match:
        # If there are previous item sections, check if the current match is after the last item section and get it
        if positions:
            if max_match_offset + max_match.start() >= positions[-1]:
                item_section = text[
                    max_match_offset + max_match.start() : max_match_offset
                    + max_match.regs[1][0]
                ]
        else:  # If there are no previous item sections, just get the text section inside that match
            item_section = text[
                max_match_offset + max_match.start() : max_match_offset
                + max_match.regs[1][0]
            ]
        # Update the list of end positions
        positions.append(max_match_offset + max_match.end() - len(max_match[1]) - 1)

    return item_section, positions


def get_last_item_section(item_index: str, text: str, positions: List[int]) -> str:
    """
    Returns the text section starting through a given item. This is useful in cases where Item 15 is the last item
    and there is no Item 16 to indicate its ending (for 10-K reports). Also, it is useful in cases like EDGAR's old
          .txt files
    (mostly before 2005), where there is no Item 15; thus, ITEM 14 is the last one there.

    Args:
        item_index (str): The index of the item/section in the report
        text (str): The whole report text
        positions (List[int]): List of the end positions of previous item sections

    Returns:
        str: All the remaining text until the end, starting from the specified item_index
    """

    # Adjust the item index pattern
    item_index_pattern = adjust_item_patterns(item_index)

    # Find all occurrences of the item/section using regex
    item_list = list(
        re.finditer(
            rf"\n[^\S\r\n]*{item_index_pattern}[.\-:\s].+?", text, flags=regex_flags
        )
    )

    item_section = ""
    for item in item_list:
        if "SIGNATURE" in item_index:
            # For SIGNATURE we want to take the last match since it can also appear in the ToC and mess up the
                  extraction
            if item != item_list[-1]:
                continue
        # Check if the item starts after the last known position
        if positions:
            if item.start() >= positions[-1]:
                # Extract the remaining text from the specified item_index
                item_section = text[item.start() :].strip()
                break
        else:
            # Extract the remaining text from the specified item_index
            item_section = text[item.start() :].strip()
            break
```

```
    return item_section


def parse_10q_parts(
    parts: List[str], text: str, items_list: List[str], ignore_matches: int = 0
) -> Tuple[Dict[str, str], List[int]]:
    """Iterate over the different parts and parse their data from the text.

    Args:
        parts (List[str]): The parts we want to parse
        text (str): The text of the document
        ignore_matches (int): Default is 0. If positive, we skip the first [value] matches. Only used for 10-Q part
              extraction.

    Returns:
        Tuple[Dict[str, str], List[int]]: The content of each part and the end-positions of the parts in the text.
    """

    texts = {}
    part_positions = []
    for i, part in enumerate(parts):
        # Find the section of the text that corresponds to the current part
        next_part = parts[i + 1 :]
        part_section, part_positions = parse_item(
            text, part, next_part, part_positions, ignore_matches
        )
        texts[part] = part_section

    return texts, part_positions


def check_10q_parts_for_bugs(
    text: str,
    texts: Dict[str, str],
    part_positions: List[int],
    filing_metadata: Dict[str, Any],
) -> Dict[str, str]:
    """Since 10-Q reports fairly often contain bugs, we check for a series of cases in this function.

    Args:
        text (str): The full text of the report
        texts (Dict[str, str]): Dictionary with the text for each part
        positions (List[int]): End-positions of the parts in the text
        filing_metadata (Dict[str, Any]): Metadata of the file

    Returns:
        texts (dict): The fixed Dictionary with the text for each part
    """

    # In some cases (mainly older .txt reports), part I is not mentioned in the text, only part II
    # Here, we can instead extract all the text before the position of part II and set it as part I
    if not part_positions or not texts:
        print(
            f"{filing_metadata['filename']} - Could not detect positions/texts of parts."
        )
    elif not texts["part_1"] and part_positions:
        print(
            f"{filing_metadata['filename']} - Detected error in part separation - No PART I found. Changing Extraction
                 to extract all text before PART II as PART I."
        )
        # The positions indicate the end of the part. So we need to substract the length of the second part to get the
             end of the first part
        texts["part_1"] = text[: part_positions[0] - len(texts["part_2"])]

    # In some cases, PART I is only mentioned in the ToC while PART II is mentioned as normal
    # Then, we would only extract the ToC content for PART I
    # By checking the distance between the two parts, we can detect this error
    elif len(part_positions) > 1:
        if part_positions[1] - len(texts["part_2"]) - part_positions[0] > 200:
            separation = part_positions[1] - len(texts["part_2"]) - part_positions[0]
            print(
                f"{filing_metadata['filename']} - Detected error in part separation - End of PART I is {separation}
                     chars from start of PART II. Changing Extraction to extract all text between the two parts."
            )
            # If the distance is very large, we instead simply extract all text between the two parts
            texts["part_1"] = text[
                part_positions[0] - len(texts["part_1"]) : part_positions[1]
                - len(texts["part_2"])
            ]

    return texts


def get_10q_parts(
    text: str, filing_metadata: Dict[str, Any], items_list: List[str]
) -> Dict[str, str]:
    """
    For 10-Q reports, we have two parts with items which can have the same name (e.g. an item 1 in part 1 and an item 1
         in part 2).
    Because of this, we need to separate the report text according to the different parts before extracting the items.

    Sometimes we get problems with the part extraction. Because of this, we check a few heuristics:
        1. If not part-texts or part-positions are found, we cannot extract anything
```

```
        2. If we don't have text for part I but have positions, we extract all text before part II as part I
        3. If the distance between part I and part II is too large, we extract all text between the two parts and add
             it to part I
        4. If part II is much longer than part I, we extract part II again but ignore the first [ignore_matches]
             matches for the
           parts until it is not much longer
        In all four cases, we raise log warnings to inform the user about potential problems in the report.

    Args:
        text (str): the full text of the report.
        filing_metadata: Dict[str, Any]: a dictionary containing the metadata of the filing.
        items_list: List[str]: list of items for the filing type.

    Returns:
        texts (Dict[str, str]): a dictionary containing the text of each part.
    """

    # Detect all existing parts in the item_list - use loop to not have duplicates but keep order
    parts = []
    for item in items_list:
        part = item.split("__")[0]
        if part not in parts:
            parts.append(part)

    texts, part_positions = parse_10q_parts(parts, text, items_list, ignore_matches=0)

    ### Check for potential problems in 10-Q reports - see docstring ###
    texts = check_10q_parts_for_bugs(text, texts, part_positions, filing_metadata)

    # In some cases, PART II already starts in the ToC & PART I only contains ToC text. PART II is then noticably
         longer than PART I
    # However, usually PART I is the much longer part.
    # In this case, we extract only PART II again but ignore the first [ignore_matches] matches for the parts until we
         find the correct PARTs
    ignore_matches = 1
    length_difference = len(texts["part_2"]) - len(texts["part_1"])
    while length_difference > 5000:
        texts, part_positions = parse_10q_parts(
            parts, text, items_list, ignore_matches=ignore_matches
        )
        # Remove text of part 1 - we will later extract all text before part 2 for part 1
        texts["part_1"] = ""

        # Check for bugs again
        texts = check_10q_parts_for_bugs(text, texts, part_positions, filing_metadata)

        # Recalculate the length difference
        new_length_difference = len(texts["part_2"]) - len(texts["part_1"])
        if new_length_difference == length_difference:
            # If the difference did not change, we stop here and extract as normal again
            texts, part_positions = parse_10q_parts(
                parts, text, items_list, ignore_matches=0
            )
            texts = check_10q_parts_for_bugs(
                text, texts, part_positions, filing_metadata
            )
            print(
                f"{filing_metadata['filename']} - Could not separate PARTs correctly. Likely PART I contains just ToC
                     content."
            )
            break
        length_difference = new_length_difference

        # If we still have a large difference, we need to ignore more matches
        ignore_matches += 1

    ### End of checking for the 4 heuristics mentioned in docstring ###

    return texts


def extract_items(
    filing_metadata: Dict[str, Any],
    items_list: List[str],
    items_to_extract: List[str],
    text: str,
) -> Any:
    """
    Extracts all items/sections for a file and writes it to a CIK_TYPE_YEAR.json file (eg. 1384400_10K_2017.json)

    Args:
        filing_metadata (Dict[str, Any]): a pandas series containing all filings metadata
        items_list: List[str]: list of all items for the filing type
        items_to_extract: List[str]: list of items to extract

    Returns:
        Any: The extracted JSON content
    """

    # Prepare the JSON content with filing metadata
    json_content = {
        "cik": filing_metadata["CIK"],
        "company": filing_metadata["Company"],
```

```
        "filing_type": filing_metadata["Type"],
        "filing_date": filing_metadata["Date"],
        "period_of_report": filing_metadata["Period of Report"],
        "sic": filing_metadata["SIC"],
        "state_of_inc": filing_metadata["State of Inc"],
        "state_location": filing_metadata["State location"],
        "fiscal_year_end": filing_metadata["Fiscal Year End"],
        "filing_html_index": filing_metadata["html_index"],
        "htm_filing_link": filing_metadata["htm_file_link"],
        "complete_text_filing_link": filing_metadata["complete_text_file_link"],
        "filename": filing_metadata["filename"],
    }

    # For 10-Qs, need to separate the text into Part 1 and Part 2
    if filing_metadata["Type"] == "10-Q":
        part_texts = get_10q_parts(text, filing_metadata, items_list)

    positions = []
    all_items_null = True
    for i, item_index in enumerate(items_list):
        next_item_list = items_list[i + 1 :]

        # If the text is divided in parts, we just take the text from the corresponding part
        if "part" in item_index:
            if i != 0:
                # We need to reset the positions to [] for each new part
                if items_list[i - 1].split("__")[0] != item_index.split("__")[0]:
                    positions = []
            text = part_texts[item_index.split("__")[0]]

            # We want to add a separate key for each full part in the JSON content, which should be placed before the
                 items of that part
            if item_index.split("__")[0] not in json_content:
                parts_text = part_texts[item_index.split("__")[0].strip()]
                json_content[item_index.split("__")[0]] = parts_text

        if "part" in items_list[i - 1] and item_index == "SIGNATURE":
            # We are working with a 10-Q but the above if-statement is not triggered
            # We can just take the detected part_text for the signature - but we do not want to run parse_item again
                 below
            item_section = part_texts[item_index]
        else:
            ### Parse each item/section and get its content and positions - For 10-K and 8-K we will just run this! ###
            item_section, positions = parse_item(
                text, item_index, next_item_list, positions, items_list
            )

        # Remove multiple lines from the item section
        item_section = item_section.strip()

        if item_index in items_to_extract:
            if item_section != "":
                all_items_null = False

            # Add the item section to the JSON content
            if item_index == "SIGNATURE":
                json_content[f"{item_index}"] = item_section
            else:
                if "part" in item_index:
                    # special naming convention for 10-Qs
                    json_content[
                        item_index.split("__")[0] + "_item_" + item_index.split("__")[1]
                    ] = item_section
                else:
                    json_content[f"item_{item_index}"] = item_section

    if all_items_null:
        print(f"\nCould not extract any item for {filing_metadata['filename']}")
        return None

    return json_content
```

Appendix B

In this Appendix, we include additional metrics for Chapter 3.

Model	**DEVELOPMENT**			**TEST**		
	μ-P	μ-R	μ-F1	μ-P	μ-R	μ-F1
SPACY	38.2 ± 0.4	58.2 ± 0.8	46.1 ± 0.1	40.8 ± 0.8	60.0 ± 0.4	48.6 ± 0.4
BILSTM (words)	78.6 ± 2.4	80.3 ± 1.2	79.4 ± 1.0	75.4 ± 1.9	78.0 ± 0.8	77.3 ± 0.6
BILSTM (subwords)	73.4 ± 0.1	77.2 ± 0.0	75.2 ± 0.1	68.8 ± 0.2	74.1 ± 1.2	71.3 ± 0.2
BERT (subwords)	74.9 ± 1.5	82.0 ± 1.8	78.2 ± 1.4	71.5 ± 1.1	79.6 ± 1.4	75.1 ± 1.1
BILSTM (words) + CRF	73.4 ± 2.0	69.3 ± 0.7	71.3 ± 1.2	70.9 ± 1.8	68.0 ± 0.9	69.4 ± 0.8
BILSTM (subwords) + CRF	80.0 ± 0.3	78.7 ± 0.5	79.3 ± 0.2	76.5 ± 0.2	76.0 ± 0.2	76.2 ± 0.2
BERT (subwords) + CRF	78.3 ± 0.3	83.6 ± 0.4	80.9 ± 0.3	75.0 ± 0.9	81.2 ± 0.2	78.0 ± 0.5

Table B.1: Entity-level micro-averaged Precision, Recall, F1-Scores ± standard deviations (3 runs) on the development and test data for our baselines.

Model	**DEVELOPMENT**			**TEST**		
	μ-P	μ-R	μ-F_1	μ-P	μ-R	μ-F_1
BILSTM (subwords)	73.4 ± 0.1	77.2 ± 0.0	75.2 ± 0.1	68.8 ± 0.2	74.1 ± 0.2	71.3 ± 0.2
BILSTM (subwords) + CRF	80.0 ± 0.3	78.7 ± 0.5	79.3 ± 0.4	76.5 ± 0.2	76.0 ± 0.2	76.2 ± 0.2
BILSTM-NUM (subwords)	77.9 ± 0.4	78.6 ± 0.7	78.2 ± 0.6	74.8 ± 0.2	76.5 ± 0.5	75.6 ± 0.3
BILSTM-SHAPE (subwords)	81.1 ± 0.1	81.5 ± 0.3	81.3 ± 0.2	77.5 ± 0.3	78.7 ± 0.5	76.8 ± 0.4

Table B.2: Entity-level micro-averaged Precision, Recall, F1-Scores ± standard deviations (3 runs) on the development and test data for the BILSTM models using the [NUM] and [SHAPE] tokens we introduced.

List of Figures

2.1 The table of contents from a randomly sampled 10-K document. The original document can be found at https://www.sec.gov/Archives/edgar/data/1326380/000162828025014731/gme-20250201.htm 11

2.2 The consolidated balance sheet of GameStop ($GME) for year 2024, as found in their original 10-K document (annual report). Retrieved from https://www.sec.gov/Archives/edgar/data/1326380/000162828025014731/gme-20250201.htm. This part comes from Item 8 of the document. . . . 12

2.3 Apart from tables and figures, financial documents also come with lots of text notes. 13

2.4 EDGAR-CRAWLER's architecture. First, the user specifies (Configuration) what data they want (companies, years, filing types) and the software downloads them. Then, the item extraction pipeline extracts the item-specific sections and converts them to a standardized JSON format. 18

2.5 Example configuration for downloading Microsoft's 10-K filings (2014-2020) and extracting items 1, 1A, 1B, 7, 7A, 8. Key parameters: skip_present_indices avoids re-downloading master index files, remove_tables strips numerical tables, skip_extracted_filings skips parsed documents. Finally, the parameters FILINGS_METADATA.csv and EXTRACTED_FILINGS coordinate the metadata across the two pipeline stages. 21

2.6 Example configuration showing how to download all 10-K filings for year 2021 from all companies, extract all items, and not removing any table. . . . 22

2.7 Example of a financial filing downloaded and parsed to a JSON format by EDGAR-CRAWLER. The original filing is a 100-page unstructed .html file (https://www.sec.gov/Archives/edgar/data/92116/000095015004000378/a97331e10vk.htm. EDGAR-CRAWLER supports multiple EDGAR filing types (10-K, 10-Q, 8-K). 23

2.8 Github stars of EDGAR-CRAWLER, across the years, as of July 2025. 24

2.9 A code fix proposed by Github Copilot for issue #20 (`https://github.com/nlpaueb/edgar-crawler/issues/20`), showing the old version in red and the new/proposed version in green, regarding the `parse_item()` function of EDGAR-CRAWLER. The change updates the regex pattern to include an additional `\n` character in the pattern (line 489, green), so that the match ends when a relevant item header is found on newline and not inline. 27

2.10 A code fix proposed by Github Copilot for issue #35 (`https://github.com/nlpaueb/edgar-crawler/issues/35`), showing the old version in red and the new/proposed version in green, regarding the `parse_item()` function of EDGAR-CRAWLER. The change updates the regex pattern to accept both "ITEM" and "ITEMS" using an optional character group (line 489, green). Apart from the code, we also updated the relevant coverage tests in the Github repository in order to reflect such scenarios, as mentioned in this issue. 28

2.11 Visualization of the EDGAR-W2V embeddings after applying dimensionality reduction with the UMAP algorithm (McInnes et al., 2018). Different colors indicate different entity types. 31

3.1 Sentences from FiNER-139, the dataset we introduce for XBRL tagging. FiNER-139 contains 1 million sentences with XBRL tags on numeric and non-numeric tokens. XBRL tags are actually XML-based and most tagged tokens are numeric. 40

3.2 Frequency distribution of the 139 XBRL tags used in this work over the entire FiNER-139 dataset. Label indices shown instead of tag names to save space. . 46

3.3 XBRL tag predictions of BERT (top), BERT + [NUM] (middle), BERT + [SHAPE] (bottom) for the same sentence. BERT tags incorrectly the amounts in red. BERT + [NUM] and BERT + [SHAPE] tag them more successfully (green indicates correct tags). 52

3.4 Hits@k results (%, average of 3 runs with different random seeds) on test data, for different k values. Standard deviations were very small and are omitted. . 58

3.5 A manually inspected sentence from FiNER-139 showing some inconsistencies in the gold XBRL tags of the auditors. The green '1' is correctly annotated with the XBRL tag *Lessee Operating Lease Term Of Contract*. The red '16' should have also been annotated with the same tag, but is not, possibly because the annotator thought the (same) tag was obvious. The orange numbers '0.1' and '6' lack XBRL annotations; they should have both been annotated as *Lessee Operating Lease Renewal Term*. We can only speculate that the auditor might not have been aware that there is an XBRL tag for lease renewal terms, in which case the recommendation engine of Section 3.11.3 might have helped. 59

3.6 Original LLM prompt used by Xie et al. (2024) to evaluate various models on the FNXL dataset (Sharma et al., 2023). 63

3.7 Our initial, slightly modified prompt inspired by Xie et al. (2024), used in small-scale tests. 63

3.8 Our enhanced LLM prompt used with Anthropic's Claude Sonnet 4 on FiNER-139. 64

3.9 Instructions appended to our (enhanced) prompt during LLM inference, using the [NUM] and [SHAPE] methodologies in the FiNER-139 experiments. The enhanced prompt can be found in Figure 3.8. When using [NUM] and [SHAPE], we also modified the 1-shot example we showed to the LLM to use the corresponding masking tokens instead of the original numeric ones. . . . 65

4.1 Class distribution of the 77 intent labels found in the training set of Banking77. Intent indices are shown instead of tag names for brevity. 75

4.2 A diagram of SetFit's two-stage training process. Retrieved from HuggingFace's blogpost: `https://huggingface.co/blog/setfit`. 77

4.3 In Stage 1, SetFit uses a Siamese network architecture to fine-tune sentence transformers on labeled pairs using contrastive loss, creating task-specific embeddings for the subsequent classification stage. 77

4.4 The prompt template we used to query the Large Language Models (LLMs). . 79

4.5 Dynamic Few-Shot Prompting: Our LLM prompt is constructed dynamically through Retrieval-Augmented Generation (RAG), using cosine similarity for in-context data selection between the test inference sample and our train samples. We use K=5, 10, 20. 87

4.6 The prompt template we used to query GPT-4 for data-augmentation in overlapping labels. We give the task and the special instruction (to pay attention to overlapping labels) to the top, then provide the data, and then remind the LLM of the task (to generate the data). 95
4.7 The Micro-F1 Score for various few-shot settings. In the Augmented Data (black) line, 3 of the examples each time (out of the 5, 10, 15, 20) belong to the representative samples that the human expert picked from the real data. . . 96

List of Tables

2.1 Descriptive statistics for the various EDGAR filings: 10-Ks (annual reports), 10-Qs (quarterly reports), and 8-Ks (current reports), which have also been the focus of prior studies (Cazier and Pfeiffer, 2016; Wallek and Ebert, 2016; Loughran and McDonald, 2016) 10

2.2 EDGAR-related mining OSS libraries, (closed-source) paid web services and data products, along with their functionalities and costs. ✓ denotes minimal support of the feature, while ✓✓ denotes extensive support. 15

2.3 Financial corpora derived from SEC (lower part) and other sources (upper part). The only works focusing on annual reports (10-K filings) are the ones from Kogan et al. (2009) and Tsai et al. (2016), where they publish only one item (Item 7) from the reports, while in our work (EDGAR-CORPUS), we collect, parse, and publish all item sections. 16

2.4 Tickers used for the manual parsing accuracy evaluation of EDGAR-CRAWLER, categorized by market capitalization for the 2023 and 2024 filing years. . . . 25

2.5 The structure of a 10-K filing with its 23 different items, grouped by section (Parts I–IV), as well as the requirement of each item per the SEC. 29

2.6 Sample words from EDGAR-W2V embeddings (top row) and their corresponding nearest neighbors (columns) based on cosine similarity. 32

2.7 Training set statistics for the FinSim-3 shared task dataset showing the distribution of financial terms across the 17 FIBO ontology categories. 33

2.8 Statistics for the different splits of the FiQA Sentiment Analysis task. 34

2.9 Results across financial NLP tasks, with different static word embeddings, as well as some LLM-derived embeddings. We report averages over 3 runs with different random seeds. 35

3.1 Examples of previous entity extraction datasets. Information about the first four from Tjong Kim Sang and De Meulder (2003); Pradhan et al. (2012); Doddington et al. (2004); Kim et al. (2003). 43

3.2 FiNER-139 statistics, using SPACY's tokenizer and the 139 tags of this work (± standard deviation). 46

3.3 Entity-level μ-F_1 and m-F_1 (%, avg. of 3 runs with different random seeds, ± std. dev.) on test data. Surprisingly, BILSTM with word embeddings (underlined) performs slightly better than BERT. Additional detailed results can be found in Appendix B. 49

3.4 Entity-level micro-averaged P, R, F_1 ± std. dev. (3 runs) on the dev. and test data for BERT-based models. Underlining denotes the best metrics within each group, while bold denotes the best metric across all different groups. . . 53

3.5 Number of total parameters (*Params*) and the best hyper-parameter values for each method, i.e., number of recurrent layers (L), number of recurrent units (U), dropout probability P_{drop}, learning rate (LR). 55

3.6 Entity-level μ-F_1 and m-F_1 (%, avg. of 3 runs with different random seeds, ± std. dev.) on test data using different ways to alleviate fragmentation. 56

3.7 Entity-level μ-F_1 and m-F_1 (%, avg. of 3 runs with different random seeds, ± std. dev.) on test data for BILSTM models with [NUM] and [SHAPE] tokens. Additional detailed results can be found in Appendix B. 57

3.8 Performance evaluation on FNXL dataset, based on Macro and Micro metrics, using the AttentionXML Pipeline. Results copied from the paper of Sharma et al. (2023). 61

3.9 Bucket analysis for benchmarked techniques on FNXL dataset. Both ours (Loukas et al., 2022) and AttentionXML pipelines incorporate all of our masking techniques and the average is reported. The reported result using our methodology leverages the BERT-base models (and not SEC-BERT). Results copied by the paper of Sharma et al. (2023). 61

3.10 The zero-shot performance of different LLMs on FNXL, according to the FinBen paper (Xie et al., 2024). All results are the average of three runs. . . . 62

3.11 Entity-level μ-F_1 and m-F_1 on test data for Anthropic's Claude Sonnet 4 models, along with the addition of [NUM] and [SHAPE] tokens during inference. . 65

4.1 Example banking intents and their labels from the Banking77 dataset. In total, there are 77 different labels in the dataset with highly overlapping semantic similarities. 70

4.2 Banking77 dataset statistics. The average lengths are shown along with their corresponding standard deviations. 74

4.3 Classification results on the Banking77 test set across all evaluated models. "N-shot × 77" denotes that N examples per class (77 intent classes in total) were used either for training (in the case of BERT-based models with SetFit) or for prompting (in the case of LLMs). For LLMs, representative samples refer to 3 examples per class selected by a domain expert, while random samples are drawn without any curation. The MPNet-v2 model is also evaluated after full fine-tuning (without SetFit) on the complete dataset (Full-Data setting) for comparison. 82

4.4 Cost analysis of various closed-source LLMs. The cost is calculated by counting the input tokens sent to the LLM. We do not count output tokens since they consist of few words, are dynamic and they are negligible. The first two groups represent the 1- and 3-shot results, multiplied by 77 classes. The rest groups come from utilizing Dynamic Few-Shot Prompting using Retrieval-Augmented Generation (RAG) for cost-effective LLM inference. Note that in the latter method, we only keep the top K=5/10/20 similar and we do not retrieve other samples. We perform 3,080 queries to each LLM, i.e., 1 query per sample in the test set. 84

4.5 Top misclassified labels, along with their misclassification percentages (out of 40 test instances), for the GPT-4 (3-shot) and GPT-3.5 models (1-shot). . . 90

4.6 Misclassification percentages (out of 40 test instances) for the MPNet-v2 model when trained on Full-Data vs. when trained on 10 samples per class. . 90

4.7 Intent recognition accuracy for retrieval+ICL and baseline methods by Milios et al. (2023). Their "retrieval+ICL" method is essentially the same one we use and call "Dynamic Few-Shot Prompting with RAG". The retrieval/training dataset size is given by the second row of the header (10-shot is 10 examples per class, 5-shot is 5). The "4K" suffix in the LLaMA 2 models means the authors there greedily filled the context windows to explore how much gain is possible if you pack in as many examples as will fit in the model input, finding that it actually degrades performance on the 7b variant but helps the larger variant (70b). 92

B.1 Entity-level micro-averaged Precision, Recall, F1-Scores ± standard deviations (3 runs) on the development and test data for our baselines. 133

B.2 Entity-level micro-averaged Precision, Recall, F1-Scores ± standard deviations (3 runs) on the development and test data for the BILSTM models using the [NUM] and [SHAPE] tokens we introduced. 133