Low-Resource Financial QA with Case-based Reasoning

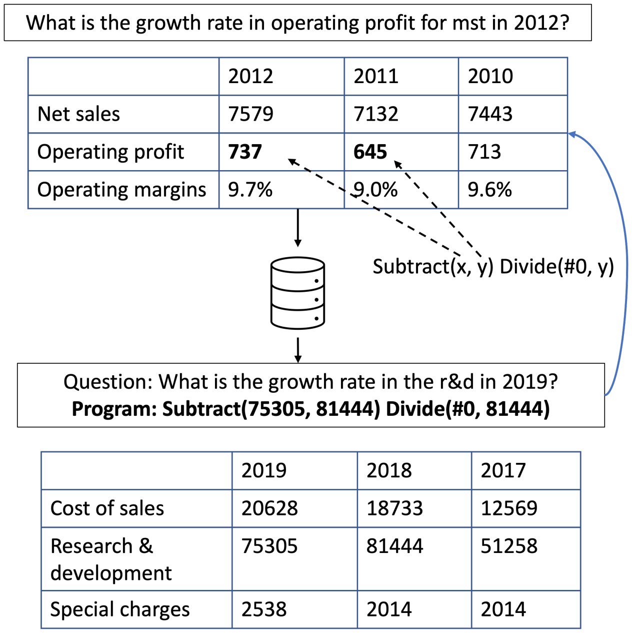
Kexuan Sun
University of Southern California
Los Angeles, USA
kexuansu@usc.edu

Jay Pujara
University of Southern California
Los Angeles, USA
jpujara@isi.edu

ABSTRACT

Financial statements are rich sources of information for market analysis and investment decisions. Most financial statements consist of unstructured text such as business descriptions and structured tables presenting numerical values of financial metrics. Recent studies showed the challenge of answering financial questions due to the difficulty of numerical reasoning over unstructured and structured data. Many novel methods have been introduced to solve this task. However, most existing approaches are data-demanding, which requires a significant amount of annotation effort. We propose a new system that answers questions with case-based reasoning (CBR) to alleviate this issue. CBR is a class of approaches that solve new problems with solutions to existing problems. We propose to leverage CBR such that annotated questions will be retrieved to provide candidate program patterns to an unseen question. The system leverages the program patterns as auxiliary knowledge to generate executable mathematical programs. Our approach decomposes the task into three sub-steps: *case selection* that provides operation steps for the question, *fact retrieval* that identifies relevant facts, and *program generation* that completes the operation steps with the retrieved facts. We conduct low-resource experiments on public financial question-answering datasets and discuss the usefulness of the system.

ACM Reference Format:
Kexuan Sun and Jay Pujara. 2023. Low-Resource Financial QA with Case-based Reasoning. In *Proceedings of The 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining RobustFin workshop (KDD '23 RobustFin Workshop).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn



What is the growth rate in operating profit for mst in 2012?

	2012	2011	2010
Net sales	7579	7132	7443
Operating profit	**737**	**645**	713
Operating margins	9.7%	9.0%	9.6%

Question: What is the growth rate in the r&d in 2019?
Program: Subtract(75305, 81444) Divide(#0, 81444)

	2019	2018	2017
Cost of sales	20628	18733	12569
Research & development	75305	81444	51258
Special charges	2538	2014	2014

Figure 1: An example of case-based reasoning for Financial QA. The two questions have different answers but require the same list of operations to produce answers.

1 INTRODUCTION

Financial statements are a common way of recording situations and activities of a company in past accounting periods. They are important sources of information for investors and analysts to understand a company or an industry for making better financial decisions. For example, the growth rate of the net revenue in the past years is useful for predicting the net revenue of the incoming year; a specific expense is useful for understanding the company's future plans; the difference in the net income between multiple companies from the same industry is important for understanding this industry. With the increasing number of companies, it becomes more and more challenging to manually review and analyze financial statements.

In recent years, with the development of machine-learning techniques, researchers started to pay attention to designing automated machine-learning models to solve knowledge-intensive problems that leverage financial statements. One popular and challenging task for financial data is Question Answering (QA). Given a financial question, an automated system is expected to either select a span of text from the financial statements or produce a program consisting of a series of mathematical operations. Most prior work [3, 10] follows the retriever-generator paradigm such that a neural retriever retrieves relevant information pieces from the statements, and a generator generates programs using the retrieved information. To achieve a good performance on program generation, machine learning models usually require a large number of annotations for all possible types of programs. However, annotating a single question demands one or multiple financial experts to review the whole financial statement, which is generally resource-consuming.

In this paper, we propose to study the financial QA task in low-resource settings. For the first time, we propose to apply a classic AI paradigm, Case-based Reasoning (CBR) [9], for our task. CBR is a class of approaches that leverage existing annotated data for

KDD '23 RobustFin Workshop, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

solving new problems. CBR has been used for many other tasks such as KBQA [4]. Figure 1 shows an example of CBR for our task. Given an unseen question, we use CBR to identify similar annotated questions and use the annotated programs as auxiliary supervision to generate programs for the new question. Our approach is also similar to demonstration-based learning [2] which is to use a few training examples in natural language prompts for better model learning.

We introduce a novel CBR-based framework that consists of a *case selector*,*fact retriever*, and *program generator*. Given an unseen question, we first use the case selector to select a program pattern from annotated examples, and then apply the fact retriever to select relevant sentences from the whole financial statements. Combining the program pattern and the relevant facts, the program generator is incorporated to generate programs that can be executed to provide final answers. In the proposed framework, program patterns serve as auxiliary supervision for better program generation.

We investigate on different case selection strategies: 1) *word-pattern co-occurrence* (the frequency of the existence of a word in the questions per program pattern), 2) *dense embedding* (the similarities between low-dimensional question embedding vectors), and 3) *two-step selection* (selecting with dense embedding and filtering with word-pattern co-occurrence matrix). We conduct our experiments on two public benchmark datasets FinQA [3] and MultiHierTT [10] involving both textual and tabular data for answering questions. **Our experiments show that a good case selector can lead to a performance boost with the same base models.**

In this paper, as the first attempt, we explore low-resource financial QA by 1) proposing to replace the retriever-generator with a novel selector-retriever-generator framework, 2) introducing different case selection strategies for providing additional supervision for program generation, and 3) investigating different strategies and conducting experiments on public benchmark datasets.

2 RELATED WORK

There are two lines of research that is closely related to this paper.

Financial QA. The task of financial QA on text and tables becomes popular in recent years. A few datasets [3, 10, 11] along with approaches were introduced to solve the problem. The datasets usually include two types of questions: *span selection* and *program generation*. The prior one identifies a span of text from financial statements (i.e.*what is the income in 2020?*), and the later one uses numerical values from the statements to produce multi-step mathematical operations which lead to the final answer (i.e.*How much did the cost increase from 2020 to 2021?*). Existing works mostly follow the retriever-generator paradigm to solve the problem. For example, Chen et al. [3] uses a neural retriever to retrieve sentences from linearized tables and text, and learn a RNN-based generator; Lei et al. [7] applies a graph-based encoder to select information pieces and a tree-based decoder to generate programs; to differentiate different types of questions from each other, Zhu et al. [11] and Zhao et al. [10] also incorporate a classifier to predict question type and learn separate generators to solve different types of questions.

Case-based Reasoning. CBR [1, 6, 9] is a class of approaches that solve new problems based on the solutions to existing problems. CBR has been used in different domains/tasks. For example, Das et al. [4] uses CBR to retrieve sub-graph patterns and answer knowledge graph-based questions. Following existing CBR techniques, our framework retrieves program patterns of annotated examples to serve as auxiliary supervision during program generation.

3 CBR FOR FINANCIAL QA

In this section, we describe the three components of the proposed system: **Case Selector**, **Fact Retriever**, and **Program Generator**.

3.1 Case Selector

Let the training set be T_{tr}. Each example $e_i \in T_{tr}$ includes a question q_i, a set of facts $F_i = \{f_{i1} \cdots f_{in}\}$ and a program p_i consisting of one or more mathematical operations over numbers. Given a test question q, the case selector aims to identify the most similar case e_k from the training set such that answers to q can be generated following mathematical steps in the program p_k. The intuition of this module is that similar questions might follow the same operation steps even though different numbers should be used to answer the question. To obtain the program pattern from the annotated program, we simply mask numbers with a special token #. For example, if an annotated program is `Add(100, 200), Subtract(#1, 500)`, the pattern to the program will be `Add(#, #), Subtract(#1, #)`. The pattern will then be used in the next steps. To retrieve such patterns, we explore the following three different approaches:

***Word-Pattern co-occurrence*.** Given the fact that many financial questions use similar words (i.e. percent, difference, etc), a natural way to select cases is to consider the frequency of the co-occurrence of each word-pattern pair. More specifically, given n training examples $Q = \{(q_1, p_1), (q_2, p_2), \cdots, (q_n, p_n)\}$ where q_i is a question and p_i is the corresponding program pattern. Each question is represented as a list of tokens (for simplicity, we split a question into words and treat each word as a token). For each token and program pattern pair, we assign a weight based on the frequency of the co-occurrence of the pair in the training data. For each token w, if it appears t_w times in questions that are answered with the pattern p, the weight of (w, p) is t_w. The weights are then normalized over patterns. Given an unseen question, based on the set of tokens appear in the question, this approach ranks all seen program patterns based on weights of the co-occurrence matrix.

***Dense embeddings*.** Similar to neural retrievers that are commonly used for open-domain question answering, another option to select cases is to use language models to represent questions and select questions based on the similarities between the embedding vectors. During inference, we create an index for all training questions based on their representations and efficiently select the most similar question as the example for each individual test question.

We use a pre-trained dense passage retriever (DPR) as the base model for representing questions. To fine-tune the model, for each candidate pattern, we randomly select a few questions as anchor questions. For each question, we then randomly sample a question with the same program pattern as a positive sample, and another question with a different pattern as a negative sample. The model is fine-tuned to make the anchor questions more similar to the positive samples than the negative samples.

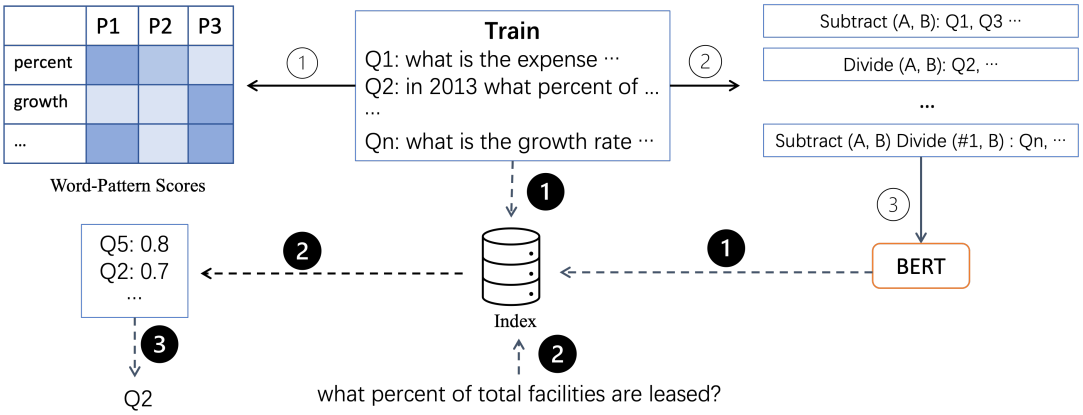

P1
P2
P3
percent
growth
...
Word-Pattern Scores
1
Train
Q1: what is the expense ···
Q2: in 2013 what percent of ...
...
Qn: what is the growth rate ···
2
Subtract (A, B): Q1, Q3 ···
Divide (A, B): Q2, ···
...
Subtract (A, B) Divide (#1, B) : Qn, ···
3
BERT
1
Index
2
Q5: 0.8
Q2: 0.7
...
3
Q2
2
what percent of total facilities are leased?

Figure 2: The case selection process. Normal and dotted arrows represent the training and inference steps, respectively. During training, the selector 1) gets word-pattern scores, 2) group questions based on their program patterns in order to create positive/negative pairs for fine-tuning, and 3) fine-tune the BERT-based model. During inference, the selector 1) index all training questions with the fine-tuned model, 2) retrieve top-k questions for a test question, and 3) filter the top-k questions with the word-pattern matrix.

Two-step Selection. In addition to the two separate approaches, we also explore a hybrid approach that leverages both dense embeddings and word-pattern co-occurrence scores. Given an unseen question, the approach first retrieves top-K_1 most similar questions with dense embeddings. In the second step, the system leverages the co-occurrence matrix to select top-K_2 patterns as a filtering set: only retrieved questions that have patterns in the top-K_2 list are kept in the final example list. The most common pattern in the final list is then passed to the generator as auxiliary supervision. Figure 2 shows the details of both the training and inference procedures.

3.2 Fact Retriever

Following previous studies, we apply a BERT-based classifier as the fact retriever. The goal of fact retrieving is to select the most important and relevant pieces of information from both tabular and textual data to answer questions. To better leverage pre-trained language models, prior approaches [3, 10] apply a linearization step for tabular data. We also use this strategy in our system. More specifically, given a table, we convert the table into facts by concatenating each cell value with the corresponding top and left attributes. For example, if the first row (top attributes) of a table has a cell "number of shares", and the first column (left attributes) of the table has a cell "granted", the corresponding cell on the same row as "granted" and the same column as "number of shares" is then be verbalized as "granted number of shares is xxx".

The question and individual facts are then concatenated and passed to a BERT-based model. The model is fine-tuned to assign the relevant facts a label "1" and irrelevant ones "0". During inference, all given facts are assigned a score, and the top-k facts are selected and will serve as part of the input to the program generator.

3.3 Program Generator

The last component leverages the outputs from the previous two components. We fine-tune a generative language model to generate final programs. The input to the model consists of the natural language question, the selected program pattern, and the retrieved facts. Let q be the question, p be the pattern and F be the set of facts. The input x to the model is:

"question: what percent of the balance was used on payments? pattern: [divide # #] context: the balance on payments is 100 context: the total balance is 200 ...".

The output y from the model is expected to be *"divide 100 200"*.

4 EXPERIMENTS

Our experiments are mainly designed to evaluate the effectiveness of CBR for the financial QA task.

4.1 Settings

We use the same fact retriever as MT2Net [10]. The retriever is a BERT-based model fine-tuned as a bi-classifier. We use RoBERTa-base as the base model and follow the default parameter settings as MT2Net. For each question and all given supporting facts, the fact retriever selects top-N facts. We use a pre-trained DPR [5] for case selection. We set the learning rate to 1e-5, batch size to 32, and train the model for 20 epochs. During inference, the neural retriever selects 50 cases, the sparse selector keeps questions with the top-20 most likely patterns, and the program generator uses the most common one among the top-10 of them as the final pattern. For program generation, we use T5-base [8] as the base model. We set the learning rate to 5e-5, batch size to 2, and train the model for 60 epochs. To evaluate our approach under low-resource scenarios, for each program pattern, we randomly sample k questions from the complete training set and use the selected questions to train all models. We set k to be 5, 10 and 20.

4.2 Datasets

We evaluate our system with two public financial QA datasets: 1)**FinQA** each example includes a question, a table and a set of texts around the table in the financial document. To answer the question,

	MultiHierTT			FinQA		
	5	10	20	5	10	20
MT2Net	0.02	0.02	0.07	0.05	0.05	0.06
Ours w/o CBR	0.02	0.07	0.10	0.08	0.10	0.16
Ours w/ CBR	**0.04**	**0.10**	**0.14**	**0.10**	**0.15**	**0.16**

Table 1: Results of different sample numbers per pattern.

	MultiHierTT			FinQA		
	5	10	20	5	10	20
Ours	0.04	0.10	0.14	0.10	0.15	0.16
Oracle	0.13	0.20	0.25	0.26	0.30	0.37

Table 3: Analysis on the effect of case selection.

	MultiHierTT			FinQA		
	5	10	20	5	10	20
Word	8.6	11.7	24.0	6.3	10.3	13.8
Dense	17.1	**27.0**	27.8	16.5	22.3	26.9
Ensemble	15.4	26.8	**37.5**	17.1	26.7	33.1
Vote	**17.2**	25.6	35.6	**18.8**	**28.7**	**34.4**

Table 2: Ablation Study on different case selection approaches.

we need to use both tabular and textual information. 2)**MultiHierTT** Each example includes a question, multiple tables with more complex hierarchies, and a set of texts around the tables.

4.3 Experimental Evaluation

We conduct the following experiments:

Experiment 1: Overall Low-resource settings w/ and w/o CBR. In this experiment, we show the effect of the retrieved patterns on the overall performance with a limited number of annotations. For each program pattern, we select 5, 10, and 20 samples, that use the pattern to answer questions, to train the model. We evaluate the model on all test questions. For FinQA, we use the public test set, and for MultiHierTT, we use the public development set. We also provide results of MT2Net as a baseline. Table 1 shows the results. Compared to both the baseline model and the variant without CBR, our approach consistently achieves better performance. The results indicate that program patterns potentially provide useful information for generating programs.

Experiment 2: Ablation Study on Case Selection Approaches. We show the case selection performance for 1) sparse selector (word-pattern co-occurrence), 2) dense retriever, 3) the ensemble approach of 1) and 2), and 4) the final selection with majority vote. We use Hit@1 as the evaluation metric: if the program patterns match, then we treat it as a hit. We report the results in Table 2. In general, the dense selector performs much better than the sparse selector. With the sparse selector as the filtering model, the ensemble approach achieves even better performance. Lastly, the majority vote also boosts the performance in most cases.

Experiment 3: Further Analysis on the Effect of Case Selector. We further conduct experiments to show the ideal performance of CBR in the same low-resource setting. We compare the end-to-end results of our automated case selector and an oracle selector in Table 3. For both datasets, the oracle selector results in significantly better performance than the default setting. These results indicate that a better case selector can greatly assist in QA performance. We hope this finding helps open up a new research direction for financial QA.

5 CONCLUSIONS

In this paper, we propose a novel approach for financial QA with case-based reasoning (CBR). Our approach uses the annotations of seen questions to provide auxiliary supervision during program generation for unseen questions. As an initial attempt, we leverage an ensemble case selection approach using both a neural retriever and a word-level sparse selector. We show the effectiveness of CBR in financial QA on two public benchmark datasets. Our experimental results indicate that better case selectors lead to better QA performance. One promising future direction is to improve the case selection procedure with rich structural information.

REFERENCES

[1] Agnar Aamodt and Enric Plaza. 2001. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7:39–59.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

[3] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. Finqa: A dataset of numerical reasoning over financial data. *Proceedings of EMNLP 2021*.

[4] Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611.

[5] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

[6] David B. Leake. 1996. *Case-Based Reasoning: Experiences, Lessons and Future Directions*.

[7] Fangyu Lei, Shizhu He, Xiang Li, Jun Zhao, and Kang Liu. 2022. Answering numerical reasoning questions in table-text hybrid contents with graph-based encoder and tree-based decoder.

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

[9] Roger C. Schank. 1983. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, USA.

[10] Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600.

[11] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 3277–3287.