Open-Source Framework for Financial LLMs – Enhanced Tokenization for Numerical Data Handling

Bilal Bounajma
ETH Zürich
Department of Mathematics
bbounajma@student.ethz.ch
21-831-144

Eliot Dugelay
ETH Zürich
Department of Computer Science
edugelay@student.ethz.ch
21-821-004

Abstract

In this paper, we propose two enhanced tokenization methods–Delimiters (DelT) and Hashtags (HashT)–to improve LLMs' handling of financial and numerical data, for financial sentiment analysis. By extending BERT tokenizer's vocabulary with domain-specific tokens and structured representations of numerical data, we fine-tune models on both Financial Phrasebank-V1.0 and a synthetic dataset. Our experiments show that HashT achieves the highest accuracy (96.5%) and lowest Expected Calibration Error ($ECE = 0.009$) on Financial Phrasebank, outperforming vanilla BERT and closely matching FinBERT. On the generated dataset, fine-tuned models outperform FinBERT, with HashT achieving an accuracy of 71.0%, while FinBERT only has an accuracy of 58.6%. These results suggest that our proposed methods help models better understand financial and numerical data, with minimal additional parameters.

1 Introduction

1.1 Financial and Open-Source LLMs

In recent years, (large) language models have demonstrated remarkable performances across a wide range of tasks OpenAI [2024], Minaee et al. [2025], Google [2023] and are now a key tool in many aspects of the tertiary sector. Finance is no exception and the use of LLMs has already facilitated several finance adjacent tasks such as coding (Jiang et al. [2024]) or information retrieval (Tang et al. [2024]).

One of the key aspects of LLMs that makes them so powerful is their versatility: they can be trained to perform specific tasks. Such LLMs are sometimes called SLLMs (Specialized Large Language Models). Famous examples are Github's Copilot (Chen et al. [2021]) specialized in generating and fixing code, Med-PaLM 2 (Singhal et al. [2023]) which can answer medical questions with very high accuracy or Chatlaw (Cui et al. [2024]), a knowledge graphs based legal assistant .

Some Financial LLMs (sometimes called FinLLMs) already exist. The most famous one is BloombergGPT (Wu et al. [2023]), a FinLLM able to answer financial questions, generate text, extract key information and conduct financial sentiment analysis. Our base paper Yang et al. [2023] however introduces a first open source FinLLM.

The "openness" of LLMS has been the subject of many discussions in the past few years (Manchanda et al. [2025], Alizadeh et al. [2024]). Models can be roughly classified into two categories: open source and closed source. The most famous open LLM would be DeepSeek (DeepSeek-AI and al. [2025]) whose entire architecture is public, whereas closed source models include the latest OpenAI's models (e.g., GPT-4 OpenAI [2024]).

Machine Learning for Finance and Complex Systems, FS2025. ETH Zürich.

However, having public architecture does not imply reproducibility and full transparency: one key aspect of LLMs' training is data retrieval and curation, and almost all public technical reports are opaque when it comes to training data, including so-called open models like DeepSeek. This has led to some discussions on the term "open LLM" with some people claiming that open LLMs should not only have open architecture but also public training procedures and data. BloombergGPT is clearly on the more closed end of this spectrum, especially given it is trained using the massive private financial data of Bloomberg.

The goal of the original paper Yang et al. [2023] is to address those two aspects altogether: provide an open-source and finance-oriented large language model.

1.2 The FinGPT framework

To build their model FinGPT, the authors proposed a general framework for open FinLLMs (Figure 1). Their pipeline is an end-to-end framework, from the data source(s) to various downstream applications in finance.

Given the massive scale of the proposed framework, it was infeasible for them (as well as for us) to use and implement each of the proposed aspects. They however manage to create models able to perform different financial natural language processing (NLP) tasks: sentiment analysis, financial report analysis, forecasting and retrieval augmented generation.

However, their base model for some of those tasks (for example Financial Report Analysis) is GPT-4, which is closed source, and they used data that is only available upon request for training, in contradiction with their open-source vision.

During this project, we also implement a part of the framework and got an end-to-end pipeline only using open models and data.

1.3 Our contribution

We propose a FinLLM that:

- Is based only on open-source models.
- Is trained only with public and free data.
- Aims to improve LLMs' handling of domain-specific terms and numerical data.

To do so, we follow the proposed framework and implement each of the four main steps: data source, data engineering, LLMs and applications. In this report, we will detail our pipeline following these steps. Since we decided to follow the order of the original paper, the explanation of some choices might be given in a later step.

The Github repository for this paper is publicly available at: GitHub Repository.

2 Data

In the past years, the increasing size of machine learning models has shed light on the importance of data to train any deep learning model (Dagès et al. [2023]). It is therefore crucial to adopt a data-centric approach.

2.1 Financial data

Modern financial data mainly comes in two forms:

- Time series representing the variation of various financial quantities: stock market prices, option prices, cryptocurrency prices and volatility, etc.
- Texts: economic, financial and political news, company fillings, social media discussions, etc.

Since we focus on NLP tasks, we focus on the latter.

One of the main challenges with financial data retrieval is that financial data is very scarce, particularly free, public data. We used the three following data sources:

- Financial Phrasebank-V1.0[1]: a dataset of 4840 English sentences for financial sentiment analysis. Each sentence is labeled (either positive, neutral or negative) by 5 to 8 financial or economic experts.
- Wikipedia to get an exhaustive list of stocks tickers.
- Yahoo! Finance.

2.2 Numerical data

One of the key observations we made is that previous models struggle with quantitative data (we will come back to this aspect). We therefore wanted to see if we could get ours to handle numerical data better. To do so, we needed quantitative financial data, which is typically the type of data that is hard to find.

To get our model to train and perform using numerical data, we decided to generate it procedurally, as it is sometimes done during pretraining of LLMs (Zhu et al. [2025]).
We did so the following way: we created sentence templates with blanks, with different sentiment connotation:

- Positive, e.g "{*company*} gained {*value*} last quarter"
- Positive/Neutral/Negative (depending on the *values*), e.g "{*company*} revenue went from {*value*} to {*value*}."
- Negative, e.g "{*company*} revenue went down of {*value*} last quarter"

Then, we fill up the first blank by a company name (e.g: "Apple", "Google", "Microsoft",...) and the remaining blanks (either one or two) by randomly generated numbers.

Since we want labeled data, we have to associate a sentiment to the sentence we have just generated. The key idea here is that the label depends not only on the sentiment of the template but also on the numerical values we have generated.
For example, the following three random sentences:

- Sony revenue went from ¥200M to ¥400M.", Label=Positive
- Sony revenue went from ¥395M to ¥400M.", Label=Neutral
- Sony revenue went from ¥500M to ¥270M.", Label=Negative

all have the same sentence template (in this case the template itself has a neutral sentiment) but have a different label, depending on the difference between the two numbers. The same idea holds for all the templates. The goal is to have a numerical dependent financial sentiment analysis dataset.

Although the use of such synthetic data is less reliable than real data fetched from financial news or stock markets (Zhang et al. [2024]), it still allows the model to understand and handle numerical data better.

3 Tokenization & Embeddings

3.1 Tokenization Process

LLMs cannot process raw text directly; it must first be converted into numerical representations. This begins with splitting the raw text into smaller strings called *tokens*, which are then mapped to numerical representations called *embeddings* (see Section 3.2).

3.1.1 Tokenization Methods in LLMs

Modern LLMs rely on subword tokenization techniques. These methods split complex words into smaller, more frequent units, enabling models to handle a wide range of text inputs.

[1] `https://www.researchgate.net/publication/251231364_FinancialPhraseBank-v10`

The most commonly used tokenization techniques include:

- **Byte-Pair Encoding (BPE)**: Iteratively merges the most frequent pairs of characters (or sequences of characters) to form subwords (Gage [1994]). It is used in GPT models such as GPT-2, GPT-3, and LLaMA.
- **WordPiece**: Similar to BPE, but optimized to maximize the language model likelihood (of the training data). Used in BERT and RoBERTa (Devlin et al. [2019]).
- **Unigram**: A probabilistic method that starts with a large vocabulary and iteratively reduces it to maximize data likelihood until reaching a fixed vocabulary size.

Each method balances vocabulary size, efficiency, and robustness. While most often effective in general domains, they can be suboptimal in specialized areas like finance.

To illustrate this, Table 1 shows how various tokenizers split financial terms and numerical data.

Example	GPT-4o	DeepSeek-R1	LLaMA-7b-hf
NVDA	NV | DA	NV | DA	N | V | DA
bullish	bull | ish	bull | ish	b | ull | ish
+$8.3M	+$ | 8 | . | 3 | M	+ | $ | 8 | . | 3 | M	+ | $ | 8 | . | 3 | M

Table 1: Tokenization of financial vocabulary examples across different LLMs, using Tiktokenizer

As one can see, financial terms and numerical expressions are often split into multiple tokens, which may reduce the model's reasoning accuracy.

3.1.2 Enhance Tokenization for Finance

Tokenization directly affects how embeddings are formed. Splitting key financial terms or symbols into suboptimal tokens, as previously seen in Table 1, can lead to poor representations. This impacts downstream tasks such as reasoning, summarization, and classification.

Financial Vocabulary

A straightforward way to improve tokenizers in the financial domain is by augmenting their knowledge of the financial vocabulary. This includes domain-specific expressions, acronyms, and stock tickers (e.g., AAPL, AMZN, NVDA). The simplest approach is to add such tokens directly to the tokenizer's vocabulary.

For better robustness, it can be wrapped in templates—for example `[STOCK:...]`—to avoid interference with terms that may already be in the tokenizer's vocabulary. Importantly, adding an entire financial dictionary is typically inefficient or impractical. Instead, the added vocabulary should be limited to the most frequent and relevant terms for the target task. For example, it may be relevant to include stock tickers like AMZN (Amazon) or major indices like the S&P500, but not necessarily those of small-cap companies.

Numerical Data

While modern public LLMs can handle many arithmetic tasks, this does not imply they truly understand or reason over numerical data. In fact, most LLMs still struggle with numerical reasoning and calibration (Boye and Moell [2025]).

To overcome these limitations, LLMs are often augmented with external tools (e.g., calculators, APIs). While generally effective, this approach introduces security risks, such as prompt injection[2]. Consequently—among other reasons—major financial institutions restrict the use of public LLMs and instead deploy their own internal models, which are typically less capable.

Improving LLMs' handling of numerical data therefore remains an active and essential area of research, particularly in finance. We present two methods that aim to improve the tokenization process for numerical handling by adding special tokens:

[2]Security vulnerability where a malicious user's input overrides the original instructions in a prompt (Schulhoff [2025])

- Delimiter tokens: Introduces domain-specific tokens: <SON> (Start of Number), <VAL> (Value), and <EON> (End of Number) to delimit numerical spans in texts:

  $$12.5 \rightarrow \texttt{<SON>2.1<VAL>12.5<EON>}$$

  where the first two digits (in the above example: '*2.1*') respectively indicate the number of digits in the integer part and decimal part of the raw value.

  These delimiters help models identify and isolate numerical spans in financial documents (e.g. news articles, SEC filings). It has been shown in Schwartz et al. [2024] to improve small LLMs' numerical reasoning performance on various arithmetic tasks.

- Hashtags template:
  1. Mask certain digits using # placeholders to preserve magnitude and unit, while allowing partial generalization.
  2. Compact and standardize format with magnitude suffixes (e.g., `K`,`M`,`B`,`T`)
  3. Replace with a structured placeholder in the format `[FinNUM:...]`.

  Examples:

  `12,500,000 dollars` $\rightarrow$ `[FinNUM:$###M]` `$12,5 millions` $\rightarrow$ `[FinNUM:$##M]`
  `-$3,5 millions` $\rightarrow$ `[FinNUM:-$#.#M]`
  `-2.56%` $\rightarrow$ `[FinNUM:-2.##%]`

  This method enables the tokenizer to handle a manageable set of numeric tokens—that is, the number of new tokens can be explicitly controlled. By masking digits, one increases generalization and reduces the number of new tokens. Conversely, by leaving digits unmasked, precision improves, but at the cost of a larger token set and reduced generalization.

  A downside is the loss of fine-grained precision, as it may not distinguish between 'similar' values (e.g., $1.1M vs $1.9M). One possible solution to this limitation would be to integrate a monotonic function, or small MLP, that takes the raw numerical value as input and produces a vector, which could then be added or concatenated to the model's embedding. This approach would be conceptually similar to the positional embeddings used in transformers.

3.2 Embeddings

Once tokens are formed, they must be converted into numerical representations, known as embeddings. This is the role of the embedding layer, which maps each token to a high-dimensional vector that serves as the model's input.

These (trained) embeddings capture semantic information, and relationships between words, allowing the model to represent meaning and context effectively. In transformer-based models, embeddings are not static: their meaning depends on the context, enabling the model to differentiate the same word in different context.

4 LLMs

Nowadays, there exists an extensive literature studying Neural Networks based LLMs. In this section, we present three aspects of LLMs and their training that we used.

4.1 LLMs alignment

After pretraining (e.g., token prediction or masked language modeling, Han et al. [2021]), a model must be aligned with human preferences. This is typically done using Reinforcement Learning with Human Feedback (RLHF), in three stages:

1. Supervised Fine-Tuning (SFT): Prompts from a dataset are paired with expert-generated answers. The model is trained via cross-entropy loss on this data.
2. Reward Modeling: The model generates outputs for prompts; experts rank them. A reward model is trained to assign higher scores to better outputs.
3. Reinforcement Learning: The model is updated via PPO (Proximal Policy Optimization, Schulman et al. [2017]) using the reward model.

The two last steps allow one to overcome a key challenge: outputs from generative tasks are hard to evaluate. In our case, however, this is not an issue. Indeed, we work with labeled data (a sentence and a sentiment), which allows us to only perform the first step, SFT. We therefore train our model via SFT on both the Financial-PhrasesV1.0 dataset and our Generated data.

4.2 Transformers

Introduced in 2017 Vaswani et al. [2023], transformers are a neural network architecture that has revolutionized LLMs. Almost all state-of-the-art models (GPT, DeepSeek, LLaMA, Claude, Mistral, etc.) are transformer-based.

This architecture consists of two main components: an encoder and a decoder. The encoder processes and understands inputs, while the decoder generates outputs. Originally designed for translation tasks, this dual structure allows for accurate understanding and high-quality generation.

Importantly, these components can be used independently: encoder-only models (e.g., BERT Devlin et al. [2019], RoBERTa Liu et al. [2019]) serve analysis tasks, while decoder-only models are mainly used for generation.

A major innovation is the multi-head self-attention, which enables the model to focus on different parts of the input per token. Each attention head captures different relationships between tokens, and their outputs are aggregated.

The encoder is composed of N blocks, each with two sub-layers: a multi-head attention layer and a feed-forward network, both wrapped with residual connections and followed by layer normalization.

The decoder is similar but adds a masked multi-head attention layer (to prevent looking at future tokens). A linear layer and softmax output are used to produce a probability distribution over the vocabulary.

4.3 LoRA Fine-tuning

Fine-tuning refers to the process of training an already trained and aligned model on some specific data or set of prompts in order to improve its performances in a given domain.

It can be done in various ways: for example, one can use full fine-tuning, i.e. retrain the whole model via the usual forward pass/back propagation method on the specific data. However, this strategy has a lot of drawbacks: it can be computationally expensive for large models, can lead to catastrophic forgetting (the model losing general knowledge it learned during the initial training which leads to very bad generalization outside of the specific task) and to usual overfitting (i.e. memorizing the training data).

It is therefore common practice to use PEFT (Parameter Efficient Fine-Tuning) methods (Xu et al. [2023]).

PEFT methods are methods during which a model, smaller than the base model, is trained. The two main advantages of such methods are that, as their name hints, they are computationally efficient (as a small number of parameters is trained) and they allow to retain most of the general original knowledge of the base model.

The trainable parameters can be a subset of the base model, for example in bias update, the weights parameters are frozen and we only update the biases (Zaken et al. [2022]). But, we can also freeze the entire base model and add new trainable parameters.

This is the idea behind adapters Hu et al. [2023]: in between some sub-layers (depending on the type of adapters) of the transformer architecture, we add a so-called adapter, a small module which will be trained while the base model remains frozen.

We use a type of adapter called LoRA (Low Rank Adaptation) Hu et al. [2021]. Those adapters are small bottleneck modules, i.e. a sub-layer of the form $x \mapsto (B \times A)x$ where $A \in \mathbb{R}^{r \times d}, B \in \mathbb{R}^{d \times r}$ where d is the internal dimension of the base model and $r < d$ is the bottleneck dimension. We add them in parallel with the FFNN sub-layers: at each feedforward sub-layer of weight matrix W and of bias b, for an input $x \in \mathbb{R}^d$, we output $Wx + (B \times A)x + b$.

We only update the A and B matrices of each FFNN sub-layer. They are usually initialized as follows: $A \sim \mathcal{N}(0, \sigma^2 I)$ with small σ (for example $\sigma = 0.02$) and $B = 0$.

We choose to use a bottleneck dimension of 8 (which is a standard).

5 Financial applications

There are various NLP tasks in finance. We introduce the one we implemented: Financial Sentiment Analysis, and briefly present a few other common ones.

5.1 Financial Sentiment Analysis

5.1.1 Definition and Uses

Sentiment analysis is a classical natural language processing task (Kumar et al. [2023]). Given text data, it aims to determine which sentiment it expresses. Usually, three sentiments are possible: Positive, Neutral or Negative.

Financial sentiment analysis is a bit more precise: the aim is to determine if a given text expresses a positive, neutral or negative sentiment on the economical and financial status of the market or company at hand (Luo and Gong [2024]).

Financial sentiment analysis has many uses:

- Stock price prediction: for example Talazadeh and Perakovic [2024] developed a sentiment-based random forest model that outperforms the standard baselines by almost 10% in averaged accuracy.
- Volatility prediction: by analyzing tweets, Wang et al. [2023] achieve state-of-the-art performances in both stock movement and volatility prediction.
- Risk prediction: in Wang et al. [2013], financial sentiment analysis is used to compare companies' risk.

We chose to implement financial sentiment analysis because it has many practical advantages compared to other tasks:

- It is a purely NLP task. LLMs and their architecture can be used for many purposes in finance, for example transformers are known for their remarkable performances for time series forecasting (Wen et al. [2023]), which can be used for stock prediction. However, we wanted to study the use of natural language processing in finance, so a text centered task seemed more appropriate.
- It is easier to train than other LLM related tasks. As stated in 4.1, a key challenge in LLM alignment is that it is hard to determine what is a good output given a input. This is why RLHF is so powerful but it requires to have an expert available to label and rank outputs, which we did not.
- For the same reason, it is easier to compare performances of different models for this task as it is to do so for other tasks.

5.1.2 FinBERT model

As mentioned in 4.2, BERT is an encoder based model. More precisely, it is made of the encoder part of the transformer architecture at the top of which a classification head is added which outputs the score of each label. By applying softmax, we obtain the probability that our input data corresponds to each label.

BERT has been adapted to various specific domains, including finance, resulting in the so-called FinBERT model Yang et al. [2020]. This model was considered state-of-the-art for financial sentiment analysis in 2021, and remains a strong baseline for such tasks today—one of current leading models being BloombergGPT.

As mentioned in 2.2, we noticed that FinBERT relies heavily on sentiment tented words and struggles with quantitative data. For example, the sentence "The company's revenue was 9 billions last quarter and is now 2 billions." is labeled as neutral with a score (i.e with confidence) > 0.99999 exhibiting both low accuracy and poor calibration. And even with sentiment tented words or symbols (like "gain", "lost","+","-", ...), FinBERT still sometimes performs poorly: "AAPL : 211.67 +13.52 +(6.82%)" is

labeled as neutral with probability > 0.96. We therefore wanted to implement a model that handles better quantitative data.

5.2 Other applications

Other financial applications of LLMs include:

- **Robo-advisory**: get personalized finance advices by using LoRA on generative models to get a specialized advisor.
- **Quantitative trading**: Generating data-driven trading signals to support strategic investment decisions.
- **Portfolio optimization**: Leveraging economic indicators and investor profiles to construct well-balanced investment portfolios.
- **Risk management**: Use NLP to identify and assess financial risk factors in text data.
- **Credit scoring**: Estimating an individual's or organization's creditworthiness based on various text data: economical news, quarterly fillings, social media posts, etc.
- **Insolvency prediction**: Forecasting the likelihood of bankruptcy based on financial statements and market trends.
- **ESG scoring**: Assessing companies' performance on environmental, social, and governance criteria through public disclosures and media.
- **Financial education**: Acting as an intelligent tutor to demystify complex financial topics and improve financial literacy.

One of the main advantages of the FinGPT framework is also to link all those applications together: for example, as mentioned above, financial sentiment analysis can be useful for quantitative trading, or risk management can be beneficial for portfolio optimization, etc.

6 Experiments

We conducted our experiments with BERT as a base model. To evaluate the impact of our two enhanced tokenization strategies, introduced in 3.1.2, we followed the pipeline below:

1. Extend the tokenizer's vocabulary with:
   - Financial vocabulary including domain-specific terms, acronyms, stock tickers, etc.
   - The three special tokens `<SON>` (Start of Number), `<VAL>` (Value), and `<EON>` (End of Number) in the case of the Delimiter tokens method (denoted as DelT)
   - The structured # tokens in the case of the Hashtags template method (denoted as HashT)
2. Train the new tokens' embeddings using the Financial Phrasebank-V1.0 dataset (on 1800 samples), while keeping the original embedding space frozen.

   Note that, when added, new tokens' embeddings are initialized randomly (typically from a normal distribution Hewitt [2021]), hence, they require targeted training to acquire meaningful semantic structure.
3. Fine-tune the model on our Generated dataset (5000 samples) using LoRA. We considered two strategies:
   - Freezing the whole model, and training only the adapter layers.
   - Training the adapter layers along with the embedding layer and classifier head.
4. Evaluate the accuracy and calibration (Pavlovic [2013]) of each obtained model, using test sets from both the Financial Phrasebank-V1.0 (450 samples) and our Generated data (1000 samples).

To analyse the performance of our models, we compare their results to the ones from FinBERT, and fine-tuned vanilla BERT.

Table 2 and Table 3 respectively show the results of our models and the FinBERT baseline on both Financial Phrasebank-V1.0 and our Generated dataset.

Model	Accuracy	Avg Confidence	ECE	#Trainable Params
FinBERT	93.6%	**99.2%**	0.38	–
Bert	73.3%	69.9%	0.08	0.40%
BertEC	76.6%	83.5%	0.08	22.1%
DelT	95.8%	94.5%	0.02	0.40%
DelTEC	95.8%	91.8%	0.05	22.3%
HashT	**96.5%**	94.9%	0.02	0.40%
HashTEC	**96.5%**	96.7%	**0.009**	22.87%

Note: EC = Unfreeze {Embedding + Classifier}; DelT = Delimiters method; HashT = Hashtags method, Avg Confidence = average softmax probability of the predicted class

Table 2: Results on Financial Phrasebank-V1.0 dataset showing accuracy, average confidence, expected calibration error (ECE), and percentage of trainable parameters per model.

Model	Accuracy	Avg Confidence	ECE	#Trainable Params
FinBERT	58.6%	**96.6%**	0.70	–
Bert	**80.3%**	61.2%	0.20	0.40%
BertEC	71.2%	80.9%	0.13	22.1%
DelT	59.4%	95.0%	0.36	0.40%
DelTEC	60.7%	93.0%	0.32	22.3%
HashT	71.0%	68.9%	**0.05**	0.40%
HashTEC	72.9%	82.8%	0.16	22.87%

Note: EC = Unfreeze {Embedding + Classifier}; DelT = Delimiters method; HashT = Hashtags method, Avg Confidence = average softmax probability of the predicted class

Table 3: Results on Generated dataset showing accuracy, average confidence, expected calibration error (ECE), and percentage of trainable parameters per model.

The results on the Financial Phrasebank-V1.0 show that both DelT and HashT methods improve over vanilla Bert, with HashT achieving the highest accuracy of 96.5% and the lowest ECE of 0.009, indicating an excellent calibration[3]. This suggests that adding these structured hashtag tokens may help the model better handle financial and numerical data. The EC variants come at the cost of significantly more trainable parameters ($0.40\% \rightarrow\sim 22\%$) but offer little improvement in the results.

As expected, on the Generated dataset, the accuracies drop, reflecting the difficulty of this dataset for typical LLMs that rely on key/tented words. Nevertheless, FinBERT's accuracy drops significantly to 58.6%, while our fine-tuned models perform better and even pretty well for some, with Bert reaching 80.3% accuracy and HashTEC 72.9%, an improvement of $+14.3\%$ compared to FinBERT.

Overall, DelT and HashT seem to be effective tokenization enhancement methods that improve model accuracy and calibration in financial sentiment tasks.

[3]A model is well-calibrated if, when it predicts a class with probability p, the true outcome happens with probability p. In finance, this is important when decisions depend on the confidence level, such as in trading or risk estimation.

References

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2025. URL https://arxiv.org/abs/2402.06196.

Google. Palm 2 technical report, 2023. URL https://arxiv.org/abs/2305.10403.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation, 2024. URL https://arxiv.org/abs/2406.00515.

Qiaoyu Tang, Jiawei Chen, Zhuoqun Li, Bowen Yu, Yaojie Lu, Cheng Fu, Haiyang Yu, Hongyu Lin, Fei Huang, Ben He, Xianpei Han, Le Sun, and Yongbin Li. Self-retrieval: End-to-end information retrieval with one large language model, 2024. URL https://arxiv.org/abs/2403.00801.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle Barral, Dale Webster, Greg S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. Towards expert-level medical question answering with large language models, 2023. URL https://arxiv.org/abs/2305.09617.

Jiaxi Cui, Munan Ning, Zongjian Li, Bohua Chen, Yang Yan, Hao Li, Bin Ling, Yonghong Tian, and Li Yuan. Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model, 2024. URL https://arxiv.org/abs/2306.16092.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance, 2023. URL https://arxiv.org/abs/2303.17564.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models, 2023. URL https://arxiv.org/abs/2306.06031.

Jiya Manchanda, Laura Boettcher, Matheus Westphalen, and Jasser Jasser. The open source advantage in large language models (llms), 2025. URL https://arxiv.org/abs/2412.12004.

Meysam Alizadeh, Maël Kubli, Zeynab Samei, Shirin Dehghani, Mohammadmasiha Zahedivafa, Juan Diego Bermeo, Maria Korobeynikova, and Fabrizio Gilardi. Open-source llms for text annotation: A practical guide for model setting and fine-tuning, 2024. URL https://arxiv.org/abs/2307.02179.

DeepSeek-AI and al. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Thomas Dagès, Laurent D. Cohen, and Alfred M. Bruckstein. A model is worth tens of thousands of examples, 2023. URL https://arxiv.org/abs/2303.10608.

Xuekai Zhu, Daixuan Cheng, Hengli Li, Kaiyan Zhang, Ermo Hua, Xingtai Lv, Ning Ding, Zhouhan Lin, Zilong Zheng, and Bowen Zhou. How to synthesize text data without model collapse?, 2025. URL https://arxiv.org/abs/2412.14689.

Jinghui Zhang, Dandan Qiao, Mochen Yang, and Qiang Wei. Regurgitative training: The value of real data in training large language models, 2024. URL https://arxiv.org/abs/2407.12835.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar 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 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Johan Boye and Birger Moell. Large language models and mathematical reasoning failures, 2025. URL https://arxiv.org/abs/2502.11574.

Sander Schulhoff. Prompt injection, 2025. URL https://learnprompting.org/docs/prompt_hacking/injection?srsltid=AfmBOoobBFy6vWEDVnunggeWjYUFLDe6ULA_YljvVPWLUF3jaTpvuBYE.

Eli Schwartz, Leshem Choshen, Joseph Shtok, Sivan Doveh, Leonid Karlinsky, and Assaf Arbelle. Numerologic: Number encoding for enhanced llms' numerical reasoning, 2024. URL https://arxiv.org/abs/2404.00459.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. Pre-trained models: Past, present and future, 2021. URL https://arxiv.org/abs/2106.07139.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL https://arxiv.org/abs/2312.12148.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022. URL https://arxiv.org/abs/2106.10199.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models, 2023. URL https://arxiv.org/abs/2304.01933.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Sudhanshu Kumar, Partha Pratim Roy, Debi Prosad Dogra, and Byung-Gyu Kim. A comprehensive review on sentiment analysis: Tasks, approaches and applications, 2023. URL https://arxiv.org/abs/2311.11250.

Wei Luo and Dihong Gong. Pre-trained large language models for financial sentiment analysis, 2024. URL https://arxiv.org/abs/2401.05215.

Saber Talazadeh and Dragan Perakovic. Sarf: Enhancing stock market prediction with sentiment-augmented random forest, 2024. URL https://arxiv.org/abs/2410.07143.

Shengkun Wang, YangXiao Bai, Taoran Ji, Kaiqun Fu, Linhan Wang, and Chang-Tien Lu. Stock movement and volatility prediction from tweets, macroeconomic factors and historical prices, 2023. URL https://arxiv.org/abs/2312.03758.

Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. Financial sentiment analysis for risk prediction. In Ruslan Mitkov and Jong C. Park, editors, *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 802–808, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing. URL https://aclanthology.org/I13-1097/.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2023. URL https://arxiv.org/abs/2202.07125.

Yi Yang, Mark Christopher Siy UY, and Allen Huang. Finbert: A pretrained language model for financial communications, 2020. URL https://arxiv.org/abs/2006.08097.

John Hewitt. Initializing new word embeddings for pretrained language models, 2021. URL https:/nlp.stanford.edu/~johnhew//vocab-expansion.html.

Maja Pavlovic. Expected calibration error (ece): A step-by-step visual explanation, 2013. URL https://towardsdatascience.com/expected-calibration-error-ece-a-step-by-step-visual-explanation-with-python-code-c3e9aa12937d#:~:text=Definition,into%20M%20equally%20spaced%20bins.

A Appendix

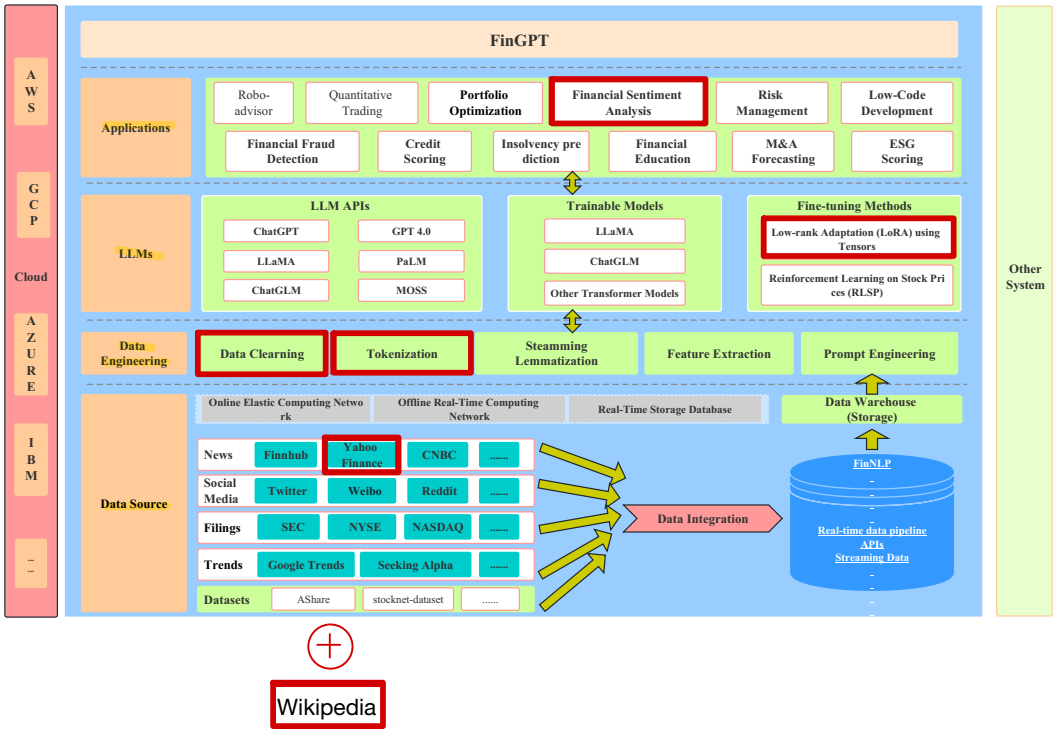
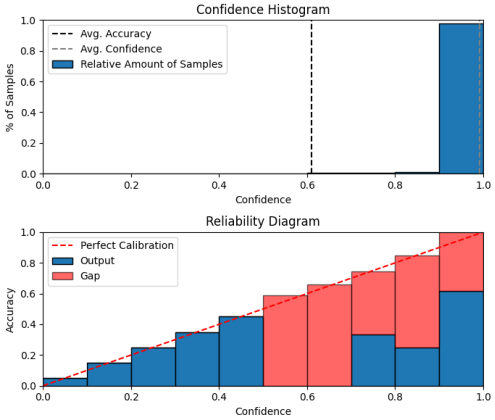
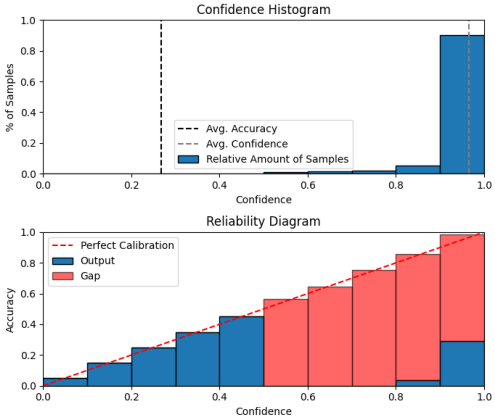


Figure 1: Overview of the FinRL-GPT Framework and our approach (in red)

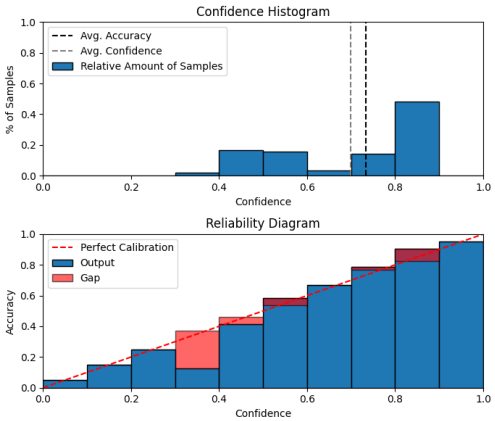
B Appendix. Reliability plots of the models: FinBERT, Bert, DeltT and HashT



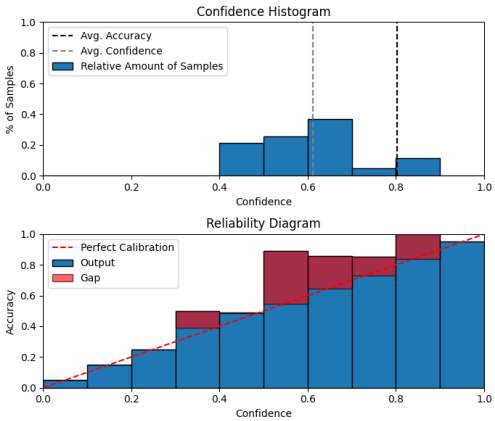
FinBERT on Financial PhrasesBank (ECE = 0.38)



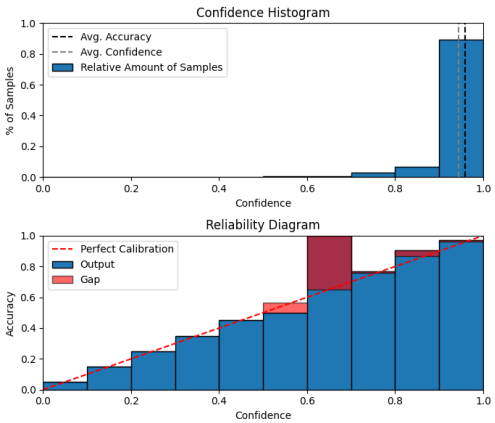
FinBERT on Generated data (ECE = 0.70)



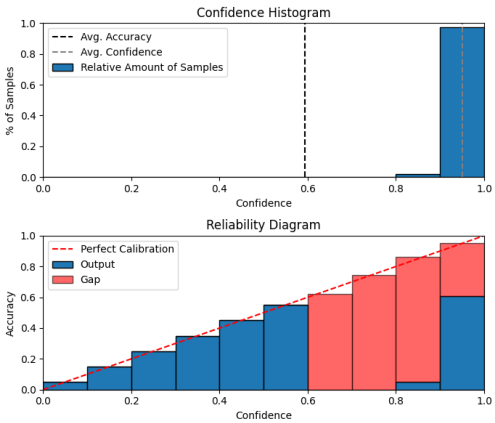
Bert on Financial PhrasesBank (ECE = 0.08)



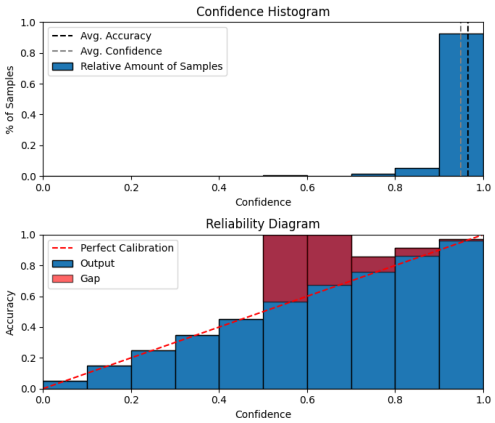
Bert on Generated data (ECE = 0.20)



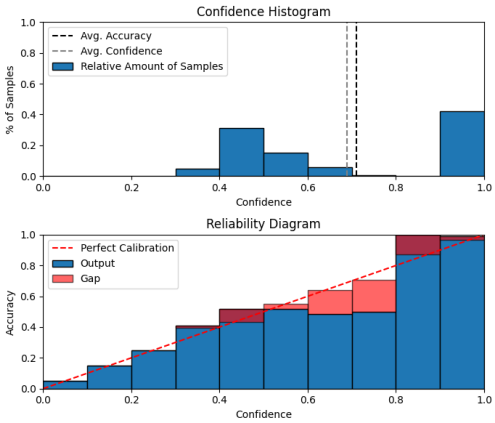
DelT on Financial PhrasesBank (ECE = 0.02)



DelT on Generated data (ECE = 0.36)



HashT on Financial PhrasesBank (ECE = 0.02)



HashT on Generated data (ECE = 0.05)

Figure 2: Reliability diagrams for some models and both datasets. Each plot compares model's confidence with actual accuracy. The closer to the diagonal, the better the calibration.