A Context-Tree-Based Algorithm for Stock Prediction and Similarity Measurement

Lubah Nelson

***Abstract*—Stock price movements are often regarded as noisy and unpredictable due to their dependence on various factors such as macroeconomic indicators, global events, and investor sentiment. However, historical price and volume data may exhibit intrinsic patterns that can guide trading strategies and enhance market understanding. This paper proposes a context-tree-based algorithm for two key tasks: predicting the next day's candlestick tuple and measuring the structural similarity of two stock charts. By quantizing price and volume data into discrete symbols and constructing probabilistic context trees, the algorithm captures temporal dependencies in the data. This approach is entirely data-driven, avoiding reliance on external datasets while ensuring robust predictions and interpretable results.**

I. Introduction

Stock price movements are frequently portrayed as noisy and influenced by countless factors—ranging from macroeconomic indicators and global news events to investor sentiment and corporate actions. Despite this apparent complexity, historical price and volume data often conceal subtle regularities. Identifying these patterns can guide trading decisions, inform risk management, and enhance overall market understanding. Over the past few decades, diverse algorithms have emerged to detect and exploit such patterns, spanning from basic technical indicators to sophisticated machine learning models that incorporate rich external datasets.

Many practitioners and theorists, however, emphasize the value of methods that rely solely on a stock's own historical record. By focusing on the time-series structure of price and volume data, these methods avoid the complexity and noise introduced by external factors. This approach aims to uncover intrinsic temporal dependencies—patterns that arise naturally within the stock's trading history. Two related challenges arise in this context:

- **Prediction (Part A):** Given a single stock's candlestick history, predict the next day's tuple of open, close, high, low, and volume without relying on external data.
- **Similarity (Part B):** Given two such histories, measure their structural similarity by comparing their internal dynamics rather than absolute price levels or external correlations.

To address these problems, this paper proposes a context-tree-based algorithm. The core insight is to recast the raw continuous data into a symbolic form that facilitates efficient modeling. By quantizing prices and volumes into discrete intervals, each day's data point is transformed into a symbol from a finite alphabet. A probabilistic context tree is then constructed to encode how future symbols depend on recent historical contexts. This structure allows the algorithm to reveal temporal patterns, providing both a predictive mechanism for the next day's price tuple and a principled method for assessing the similarity of two given charts.

The proposed approach is inherently data-driven, requiring no external information or large training sets. It handles sparse data gracefully through pruning and fallback strategies and allows flexible quantization choices to balance granularity with statistical reliability. Furthermore, mapping predicted symbols back into continuous price and volume values ensures that the model's forecasts remain interpretable and applicable in real-world scenarios.

II. Problem Definition and Challenges

We are given a historical candlestick chart for a stock, represented by a sequence of days:

$$\alpha[1], \alpha[2], \ldots, \alpha[k],$$

where $k \geq 2000$. For each day i, the candlestick data is defined as:

$$\alpha[i] = (O_i, C_i, H_i, L_i, V_i),$$

where:

- O_i: Opening price of day i,
- C_i: Closing price of day i,
- H_i: Highest price of day i,
- L_i: Lowest price of day i,
- V_i: Volume of shares traded on day i.

A. Task A: Prediction

Given $\alpha[1 \ldots k]$, the goal is to predict the tuple $\alpha[k+1]$. Unlike methods that rely on external signals or advanced machine learning models trained on large datasets, our objective is to derive a prediction solely based on the information contained in the given chart. This presents a challenge: how to effectively capture temporal patterns from a single long sequence without introducing oversimplified assumptions or requiring external market indicators.

B. Task B: Similarity

Given two charts $\alpha[1 \ldots k]$ and $\beta[1 \ldots k]$, each represented by their own daily tuples, the objective is to measure their similarity. The notion of similarity should reflect how the two sequences evolve over time, rather than relying solely on simple correlations. The method must compare their internal probabilistic structures—examining how likely certain patterns are, and how contexts lead to specific price/volume configurations—instead of raw price levels or external benchmarks.

C. Key Challenges

1) **No External Factors:** The algorithm must extract and leverage patterns purely from the input candlestick arrays, without using macroeconomic data, sentiment analysis, or industry benchmarks.
2) **Sequential Dependency:** Stock prices are inherently time-dependent. Patterns such as trends, mean reversions, or volatility clusters require models that respect the temporal order and conditional structure of the data. A suitable algorithm must encode how today's prices and volumes relate to recent days and how these patterns may recur.
3) **Efficiency with Large k:** For $k \geq 2000$, the algorithm must efficiently handle large datasets. It must avoid explosive complexity and excessive memory usage, even when encoding multi-dimensional data into symbolic form and constructing complex data structures.
4) **Complexity of Continuous Data:** The input consists of real-valued price and volume data. Directly modeling such continuous values is challenging, especially when aiming to construct a probabilistic model of sequences. Symbolizing the data (discretizing continuous ranges into symbols) simplifies the problem but raises questions about how to choose intervals, handle uneven distributions, and ensure meaningful granularity.
5) **Robustness to Noise and Rare Patterns:** Financial data is often noisy and contains rare or anomalous patterns. The algorithm must be robust, employing fallback strategies when certain contexts are too sparse and pruning rarely encountered states that contribute little predictive value.

D. Proposed Solution

The proposed solution, **based on context trees**, addresses these challenges by converting continuous data into symbols and modeling the conditional probabilities of future outcomes given historical contexts. Context trees naturally handle sequential dependencies and can be pruned or adjusted to account for noise and rare events. By carefully choosing **discretization** intervals and context lengths, as well as introducing fallback strategies, we ensure that the method scales and remains robust, even when working with a single input chart or comparing two charts without external references.

III. Intuition Behind the Methodology

The proposed context-tree-based algorithm for stock prediction and similarity measurement can be understood through an intuitive walk-through of its two primary components: prediction and similarity. This section breaks down these ideas into simple, relatable steps.

A. Part A: Prediction of Stock Data

The Setup: We start with a long record of daily stock trading data, containing five key numbers for each day: the opening price, closing price, highest price, lowest price, and trading volume. The goal is to predict tomorrow's data based solely on patterns observed in the past days.

Handling raw numerical data can be complex, so we first convert each day's data into symbols. By dividing the range of each number into intervals and assigning labels (e.g., 'A', 'B', 'C'), each day is transformed into a sequence of symbols like 'BACAL.' This simplifies the data into a format that's easier to analyze.

Once the data is symbolized, we look for patterns in how these symbols follow each other. For example, we might notice that after the sequence 'BAC', the next day's symbol is often 'C'. These patterns are stored in a structure called a *context tree*, which captures the likelihood of certain symbols appearing after specific sequences.

To predict tomorrow's data, we examine the most recent days, find their pattern in the context tree, and use the stored probabilities to predict the next symbol. This symbol is then converted back into numbers using the midpoints of the original intervals.

B. Part B: Measuring Similarity Between Two Stocks

The Setup: Now consider two stocks, each with its own history of daily data. The task is to determine how similar their behaviors are over time, not by comparing raw prices, but by analyzing the patterns in their movements.

For each stock, we follow the same steps as in prediction: convert the daily data into symbols and build a context tree that stores the probabilities of symbols following certain patterns.

To measure similarity, we compare the probabilities stored in the two trees. For example, if both stocks' trees indicate that after 'BAC', the next symbol is likely 'C', this suggests similar behavior. The degree of similarity is quantified using a statistical measure called *Jensen-Shannon Divergence* (JSD), which calculates how closely the probabilities match.

High similarity scores indicate that the two stocks exhibit comparable patterns, which could imply shared underlying dynamics, such as reacting similarly to market conditions.

Methodology

Our approach consists of three main steps: transforming continuous candlestick data into symbols, building a context tree that encodes sequential dependencies, and applying this structure to prediction and similarity tasks. By recasting raw, high-dimensional data into discrete symbolic sequences, the methodology balances the complexity of modeling financial time series with computational efficiency, leveraging the probabilistic context tree to uncover and utilize temporal patterns.

Symbolization of Candlestick Data

Symbolization is the process of mapping each day's market data tuple $(O_i, C_i, H_i, L_i, V_i)$—representing open, close, high, low prices, and volume—into a discrete symbol. This is a necessary step because directly modeling the joint distributions in continuous space is computationally challenging and often prone to overfitting or sparsity issues.

By discretizing the data, we transform the problem into one of sequence modeling over a finite alphabet.

The process begins by dividing the range of each dimension into intervals. For example, if the close price ranges from \$100 to \$200, it can be divided into intervals such as:

$$[100, 120) = A,$$
$$[120, 140) = B,$$
$$[140, 160) = C,$$
$$[160, 180) = D,$$
$$[180, 200) = E.$$

Each interval is assigned a unique label, and for a given day's tuple, the labels from all dimensions are concatenated to form a composite symbol. For instance, if a day's open price falls into $[100, 120)$, close into $[120, 140)$, and volume is categorized as low, the resulting symbol might be `ABL`.

a) Adaptive Quantization:: While uniform intervals are straightforward and simple to implement, they may fail to capture the nuances of non-uniform data distributions common in financial time series. Adaptive methods, such as quantile-based intervals, address this by ensuring that each interval contains a roughly equal number of data points, allowing finer distinctions in dense regions and broader bins in sparse ones. For example, a stock with most prices between \$120 and \$140 might use narrower intervals in this range, while extending intervals for outlier prices. This ensures the symbolic representation reflects the underlying structure of the data without introducing sparsity or loss of critical detail.

C. Building the Context Tree

What is a Context Tree? A context tree is a data structure that represents the conditional distribution of future symbols given a finite history of past symbols. Each node corresponds to a context (a sequence of up to d past symbols), and the node stores frequency counts of what next symbols were observed after that context.

By normalizing these counts, we obtain probabilities:

$$P(s_{t+1} = x \mid s_{t-d+1}, \ldots, s_t).$$

Algorithm for Context Tree Construction:

1) **Maximum Context Length d:** Decide how many past days to consider. For instance, $d = 5$ might capture patterns spanning a week of trading.
2) **Insertion of Patterns:** For each position i in the symbolized sequence $s_1, s_2, \ldots, s_k$:
   - Extract contexts of lengths 1 up to d ending at s_i.
   - Insert these contexts into the tree, updating frequency counts for the next symbol s_{i+1}.

Pruning is closely tied to the concept of managing rare contexts. When certain contexts appear infrequently, they contribute little to the overall statistical reliability of predictions. For instance, if a node represents a context that has occurred only twice in 2000 days, it may lack the strength to provide meaningful insights. By pruning such nodes and merging them into shorter contexts, we can save memory and reduce noise, thereby ensuring the resulting probabilities are more stable and reliable.

To handle cases where deep contexts are either unavailable or too sparse, a fallback strategy comes into play. If a d-length context is not found or lacks sufficient data, we iteratively try shorter contexts, such as $d-1$ symbols, $d-2$, and so on, until we encounter a node with adequate information. This approach mitigates overfitting to rare patterns and enhances prediction reliability, even in noisy or sparse data scenarios.

PART A: PREDICTION ALGORITHM

Goal: Given $\alpha[1...k]$, predict $\alpha[k+1]$. The algorithm relies on a context tree built from a discretized symbolization of historical data.

Algorithm Steps

a) 1. Symbolization::

- Determine intervals for each dimension (Open, Close, High, Low, Volume).
- Convert each day's tuple $(O_i, C_i, H_i, L_i, V_i)$ into a discrete symbol s_i.
- **Example:**
  - Suppose after analyzing the entire history, we decide on the following intervals for the Close price (just one dimension as an illustration):

$$[100, 120) = A,$$
$$[120, 140) = B,$$
$$[140, 160) = C,$$
$$[160, 180) = D.$$

  - If on day i, the Close price $C_i = 145$, it falls into the $[140, 160)$ range, so that dimension's symbol is 'C'. Similarly, do this for Open, High, Low, and Volume, and combine them into one composite symbol, e.g., 'BBCLL' or a code representing that combination.
- After this step, we have a sequence $s_1, s_2, \ldots, s_k$.

b) 2. Build the Context Tree::

- Choose a maximum context length d (e.g., 3 or 5).
- For each position i in s (from 1 to $k-1$), consider the d-symbol history before $i+1$. Count how often each next symbol appears following that history.
- Store these counts in a tree structure, where each node represents a context (a sequence of symbols) and has probability distributions over possible next symbols.

c) 3. Finding the Next-Day Prediction::

- Given $\alpha[1...k]$ symbolized as $s_1, \ldots, s_k$, identify the last d-symbol context: $(s_{k-d+1}, \ldots, s_k)$.
- Traverse the context tree following the path for $(s_{k-d+1}, \ldots, s_k)$.
- If a node for this exact context exists and has sufficient data (non-sparse counts), use its distribution.
- If not found or too sparse, drop one symbol from the left and use $(s_{k-d+2}, \ldots, s_k)$, and so forth, until you find a suitable context.

d) 4. Selecting the Next Symbol::

- Suppose the context node you end up at has counts:

$$\text{Counts} = \{X : 10, Y : 4, Z : 6\}.$$

- Normalize to probabilities:

$$P(X) = 0.5, \quad P(Y) = 0.2, \quad P(Z) = 0.3.$$

- Choose the symbol with the highest probability (in this case, X).

e) 5. Convert Symbol Back to Numeric Prediction::

- Each symbol corresponds to intervals for (O, C, H, L, V).
- For the chosen symbol, take the midpoint of each interval as the predicted value.
- **Example:** If the chosen symbol's Close price interval is $[140, 160)$, use the midpoint 150 as the predicted Close. If Volume's chosen interval is $[1,000,000, 1,100,000)$, use 1,050,000 as the predicted Volume.
- Alternatively, you could store all historical days that produced the chosen symbol and average their actual values rather than just taking the midpoint for a more data-driven reconstruction.

Worked-Out Example for Part A

Suppose we have daily Close prices over 10 days for a single stock. This example focuses on the Close price dimension.

Algorithm 1 Example of Symbolization and Context-Based Prediction

Input: Daily Close Prices:
- $[105, 112, 118, 145,$
- $150, 159, 162, 155,$
- $142, 149]$

Quantization Intervals: $[100, 120) \leftarrow A$, $[120, 140) \leftarrow B$, $[140, 160) \leftarrow C$, $[160, 180) \leftarrow D$

Symbolization:

for $i = 1$ to 10 **do**
  if $100 \leq \text{Price}_i < 120$ **then**
    $\text{Symbol}_i \leftarrow A$
  else if $120 \leq \text{Price}_i < 140$ **then**
    $\text{Symbol}_i \leftarrow B$
  else if $140 \leq \text{Price}_i < 160$ **then**
    $\text{Symbol}_i \leftarrow C$
  else
    $\text{Symbol}_i \leftarrow D$
  end if
end for

Symbol Sequence: $[\text{Symbol}_1, \text{Symbol}_2, \ldots, \text{Symbol}_{10}] = [A, A, A, C, C, C, D, C, C, C]$

Context Length: $d = 3$

Example Context (before Day 5): (A, A, A)

Context Lookup (for Day 11): Assume the last context before Day 11 is (C, C, C)

Historical Occurrences of Next Symbols:
- After (C, C, C): 'D' (1 time), 'C' (2 times)

Probabilities:
- $P(D|CCC) = \frac{1}{3} \approx 0.33$
- $P(C|CCC) = \frac{2}{3} \approx 0.67$

Predicted Symbol (Day 11): C (most likely)

Predicted Close Price (Day 11): Midpoint of interval C: 150

This is, of course, a simplified example focusing only on the Close price. In a real scenario, you would use a composite symbol formed from (O, C, H, L, V) intervals, providing a full predicted tuple.

PART B: SIMILARITY MEASUREMENT ALGORITHM

Goal: Given two stock charts α and β, measure how similar their underlying patterns are.

Algorithm Steps

f) 1. Normalize and Symbolize Both Charts::

- If α and β differ in scale, normalize them to a common range (e.g., map both to $[0, 1]$ for price dimensions and scale volumes similarly).
- Use identical intervals for symbolization for both α and β.
- Convert α into symbols $s_1^\alpha, \ldots, s_k^\alpha$ and β into symbols $s_1^\beta, \ldots, s_k^\beta$.

g) 2. Build Context Trees::

- Build T_α using α's symbol sequence.
- Build T_β using β's symbol sequence.

h) 3. Compare Probability Distributions::

- For each context in T_α (up to length d), retrieve the probability distribution of next symbols from T_α and, if it exists, from T_β.
- Compute a divergence measure between the two distributions, such as the Jensen-Shannon Divergence (JSD).

i) 4. Jensen-Shannon Divergence Calculation::

- If P_α is the distribution from T_α at a given context, and P_β is the corresponding distribution from T_β, define:

$$M = \frac{P_\alpha + P_\beta}{2}.$$

- Compute:

$$JSD(P_\alpha, P_\beta) = \frac{1}{2}KL(P_\alpha||M) + \frac{1}{2}KL(P_\beta||M),$$

where KL is the Kullback-Leibler divergence.

j) 5. Aggregate Similarity::

- Average the JSD values over multiple contexts or take a weighted average.
- The final similarity score could be defined as:

$$\text{Similarity Score} = \exp(-\lambda \cdot \overline{JSD}),$$

where $\overline{JSD}$ is the mean JSD. A lower $\overline{JSD}$ means more similar distributions, hence more similar patterns.

Worked-Out Example for Part B

Suppose α and β are both normalized and symbolized with the same intervals. Consider a single context: (C, C) in T_α.

Algorithm 2 JSD Calculation Example

1: **Given:** Context (C, C)
2: For T_α: $P_\alpha = \{C : 0.5, D : 0.5\}$
3: For T_β: $P_\beta = \{C : 0.4, D : 0.6\}$
4: Compute $M = \frac{P_\alpha + P_\beta}{2}$:
   - $M(C) = \frac{0.5+0.4}{2} = 0.45$
   - $M(D) = \frac{0.5+0.6}{2} = 0.55$
5: Compute $KL(P_\alpha||M)$: $KL(P_\alpha||M) = 0.5 \log\left(\frac{0.5}{0.45}\right) + 0.5 \log\left(\frac{0.5}{0.55}\right)$
6: Compute $KL(P_\beta||M)$
7: Compute $JSD(P_\alpha, P_\beta) = \frac{1}{2}KL(P_\alpha||M) + \frac{1}{2}KL(P_\beta||M)$
8: Repeat for multiple contexts and average.

Having outlined the methodology, we now address practical issues: computational complexity choosing parameters such as the number of intervals and context length d, dealing with noise and regime changes, and strategies for validating the approach on real data.,

D. Efficiency and Complexity

The computational efficiency of the proposed method is crucial for practical application. The time complexity of context tree construction involves inserting contexts for each position i in the symbol sequence. With a maximum context length d, up to d contexts might be inserted at each i, resulting in a complexity of approximately $O(k \cdot d)$, where k is the number of days. Prediction involves a single tree lookup with $O(d)$ complexity. Similarity computation requires comparing distributions at various contexts; extracting these distributions and computing the Jensen-Shannon Divergence (JSD) over selected contexts also scales linearly with the number of relevant contexts, which remains manageable for $k \approx 2000$. This polynomial complexity is feasible for moderate values of k (e.g., 2000 to 5000), which is within the common historical length of many stock datasets. Memory usage can become a concern if the number of intervals is large and d is high, leading to a large tree. To mitigate this, we employ pruning strategies. After building the tree, nodes representing extremely rare contexts (e.g., contexts that occurred fewer than 3 times) can be removed to significantly reduce the memory footprint.

E. Parameter Tuning

Several parameters influence the performance of the algorithm and require careful tuning. The number of intervals used for quantization represents a trade-off: fewer intervals result in a coarser representation, fewer symbols, and a lower risk of data sparsity, but also less detail. Conversely, more intervals provide finer granularity and potentially more insight but lead to sparser data and a larger tree. As a guideline, we suggest starting with around 10 intervals for each price dimension (O, C, H, L) and 5 for volume. These values can be adjusted based on empirical results. If predictions are too coarse, the number of intervals should be increased; if the tree becomes too large or too sparse, it should be reduced.

The context length d also presents a trade-off. A smaller d captures only short-term patterns and results in more stable distributions, while a larger d can capture longer-term patterns but increases the chance of data sparsity. We recommend starting with $d = 5$. If short-term patterns dominate the data, a smaller value like $d = 3$ may suffice. If longer cycles (e.g., weekly patterns) are suspected, a larger value like $d = 7$ should be considered. Experimentation with different d values on a validation subset is essential to determine the optimal value.

Finally, pruning thresholds determine how aggressively rare contexts are removed from the tree. A frequency threshold (e.g., pruning contexts occurring fewer than 3 times) can be set to balance detail and statistical reliability. This threshold should also be tuned based on empirical results.

F. Handling Noise, Regime Changes, and Unseen Patterns

Financial markets are inherently noisy and non-stationary. Market conditions change over time, and patterns that were effective in one period may not hold in another. Drastic regime shifts (e.g., from a bull market to a sudden crash) can cause previously common contexts to become rare. The proposed method addresses these challenges through fallback and pruning strategies. The fallback mechanism ensures that the model does not rely on non-existent deep contexts. If a complex pattern disappears, the model will

naturally fall back to simpler, more stable patterns. Furthermore, rebuilding or updating the tree periodically (e.g., every quarter) to reflect recent market conditions can ensure that the model remains relevant as patterns evolve.

G. Conclusion

We have introduced a context-tree-based algorithm for analyzing stock chart data, uniquely focusing on the internal temporal dependencies within price and volume time series. This data-driven approach, requiring no external inputs, offers a distinct perspective compared to methods relying on macroeconomic indicators or extensive training datasets. While not aiming to provide definitive market predictions, the proposed method provides a robust and interpretable framework for identifying structural similarities between stock charts and extracting probabilistic insights for short-term forecasting.

REFERENCES

[1] P. Buhlmann and A. J. Wyner. "Variable length Markov chains". In: *The Annals of Statistics* (1999), pp. 2480–2507.

[2] D. M. Endres and J. E. Schindelin. "A new metric for probability distributions". In: *IEEE Transactions on Information theory* 49.7 (2003), pp. 1858–1860.

[3] J. D. Hamilton. *Time series analysis*. Princeton university press, 1994.

[4] E. Keogh and M. J. Pazzani. "Relevance feedback for time series data mining". In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. 2001, pp. 219–226.

[5] J. Lin et al. "Finding local motifs in large time series". In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 561–570.

[6] R. S. Tsay. *Analysis of financial time series*. John Wiley & Sons, 2005.

[7] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. "The context-tree weighting method: Basic properties". In: *IEEE Transactions on Information Theory* 41.3 (1995), pp. 653–664.