A brief research in multi-agent system variants and case-based reasoning for fully automated algorithmic trading

Jesper Lim Jun Hao (A0248996B) e0934611@u.nus.edu
Pitchappan Pitchappan Ramaswamy (A0236575W) e0740930@u.nus.edu
Seow Kah Yong (A0032747E) e0383424@u.nus.edu
Zhu Yexin (A0083629W) e0696677@u.nus.edu

Abstract: Stock market is one of the most popular investing options that many people jump into expecting high profits. Information from various sources such as an event, comments, news, and many other information from the web would have an astonishing effect on the stock prices. It became critical for investors to collect, filter, access and integrate all these information to support their decision making. Also, with the advancement of technology and availability, information that would have a dramatic effect on stock prices are increasing at an astonishing rate as time goes. Therefore, it is difficult for individuals to constantly keep track and keep up to date on all this information to make an informed decision to trade. Nevertheless, using a multi-agent system in short, MAS, would automate most of the process to support decision making. Each agent has well-defined tasks such as information collecting, filtering, data conversion, mathematical calculation and decision making model. In this paper, we are going to identify some critical agents that are required and how information is being communicated and exchanged to gain knowledge to facilitate our decision making in making a trade. Moreover, we will be looking at the various CBR techniques and in the last section, we will present our preliminary proposal for a MAS for algorithmic trading where the user does not have to lift a finger in order to make profits/cut losses via automated trades.

1. Introduction: A brief look of various multi-agent systems (MAS)

There are two basic structures of MAS: Centralized and Decentralized. A centralized MAS has a centralized controller through which every agent can communicate via while a decentralized MAS basically allows every agent to communicate directly with each other without any supervisory control.[1]

There are various advantages and disadvantages when it comes to the choice of the MAS that is being chosen here.

For centralized MAS, it is more advantageous when less infrastructure is required for communication while decentralized MAS requires more infrastructure for communication. Since a centralized MAS requires a supervisory center to work, it can become a single point of failure if the controller goes offline but for a decentralized MAS, it works continuously.

Regardless the type of MAS that is being in used here, the general benefits of using a MAS are:

1. Allows us to construct a robust,flexible and extensible system.
2. A MAS fits in a distributed architecture that relies on local information and decision making.

[1] Renuka,K., Priyanka,P. & Yogendra,K. (2018). A State of Art Review on Various Aspects of Multi-Agent System, Journal of Circuits, Systems, and Computers

3. Flexible where you can just ‘plug and play’ capabilities to change the system (E.g. change an agent to another by redirecting the input and output from that original agent to the other).
4. A MAS can be resilient where it can quickly respond and adjust to faults.

Moreover, MAS has its own limitations:

1. It can result in unpredictable outcomes due to emergent behavior. This is more true in decentralized MAS where the systems can self-organize themselves.
2. Hardware implementations of MAS can be a challenge since most MAS systems currently are software based.
3. The ability for MAS systems to scale up due to increases in problem dimensions is not a well understood topic.
4. Security is an issue as each individual agent can be an attack vector and can be exposed for security breach.

Not all MAS architectures are the same, even if the application is solely meant for trading systems.

Elhadi & Saad (2005) proposed a peer-to-peer MAS system for a trading system where the aim is to achieve a win-win scenario rather than focusing on just a single attribute-based negotiation. The architecture follows more of a decentralized model where buyers and sellers find each other (rather than having traders joining markets first) via a gate-way using the interaction between agent and retrieval agents. The architecture (Appendix A) includes only three types of agents: interface agents that reside on every host machine, keep track of the user profile, interacts with other agents, spawns and initializes retrieval agents; resource agents that acts as a gate-way, responsible for local database management; retrieval agents which are responsible for communicating and interacting with other agents in remote hosts.

What makes this architecture distinctive from its peers is the implementation of its negotiation module. As the paper has stated, there are two forms of negotiation: distributive and integrative. Distributive negotiation is something that most people are familiar with, especially in the subject matter of trading where agents are self-interested and this typically results in a zero-sum game where a price tug-of-war always happens. In the context of integrative negotiation, it involves resolving a conflict among two or more parties where agents become co-operative and the negotiation is based on multiple attributes rather than price alone.

While this model does not fit well into a day-trading context, it is very conducive for long position holders who are more into the fundamentals of a particular financial product rather than the price action of the asset.

A more typical MAS architecture that is suited for trading is proposed by Tirea, M; Tandau, I; Negru, V (2011). This centralized model (see Appendix B) that is proposed contains three agents (I.e. Fundamental Analysis Agent (FAA), Technical Analysis Agent (TAA) and Prediction Agent(PA)) that generates buy/sell signals. In addition, it contains a Coordinator agent (CA) that coordinates the actions of these three agents and provides a weightage for each of these agents (E.g. 30-35-35 for PA-FAA-TAA) for its own prediction. This resembles an ensemble where the weights are used to drive a decision for the coordinator agent so the coordinator agent serves as a decision making agent as well.

One additional interesting discovery is that the PA adopts a posteriori knowledge model by using a neural network to search for patterns in the stock database which contains current security

historical data. If such patterns are found, the PA will be able to generate buy/sell signals for the CA to make a decision on. This is a form of backward chaining communication between the coordination agent, stock database and the three agents TAA,FAA & PA.

Another note-worthy takeaway from this paper is their use of various technical analysis indicators like RSI, Williams's %R, moving average and Bollinger bands which yield quite a bit of profit and the metrics that they used for fundamental analysis (I.e. PER,ROE,EPS,MO,MN).

Another rather more promising proposal is Luo, Y. & Davis, Darryl N. & Liu, Kecheng (2008)'s MASST Framework which includes even more components compared to Tirea, M., Tandau, I; Negru, V (2011)'s paper. (see Appendix C)

The model introduces a new risk management agent (RMA) that, on the basis of the user profile, interacts with the monitoring agent and decision-making agent to analyze the risk levels of user's share holdings, report the profit status and suggest a stop-loss level for the holding shares. The monitoring agent (MA) within this model also monitors the stocks that the users' are interested in using technical indicators and notifies the user if there is any abnormality in trading volume or price. The decision making agent also has an additional function that allows it to identify any activities of institutional investors in the given shares. Moreover, this model does not use a prediction model that looks for patterns in its own prediction outcomes (case-based reasoning - CBR) unlike Tirea, M; Tandau, I; Negru, V (2011)'s model which utilizes a neural network to do that.

Given the pervasive adoption of the MASST framework in automated stock trading, we look into case-based reasoning next as such information is sparse in our research for MAS and we feel that it is of great value that patterns can be recognized from past predictions made by these models.

2. Case-based reasoning (CBR)

CBR is a decision making approach where decisions that need to be made can be based on relying on previous decisions and their outcomes so that the outcome of the new decisions made can have a positive outcome in a more probable sense. These new outcomes and decisions are further added into the knowledge base to be used in the same manner for decisions to come. In a way, we are classifying these decisions (or data-points) as desirable or not. Or in the context of stock trading, we can classify if said signal is a good one (I.e. true positive/negative) or bad one (I.e. false positive/negative). It is only natural for us to look into the different classifiers that can serve this purpose.

Before we delve into the topic of classifiers for retrieval, we look at the main aspects of CBR.

Giorgio, L., Luigi,P., Paolo A. & Marco,V. (2016) covers a few important points about CBR. Briefly, CBR is a lazy learning technique where we keep training instances directly in memory and we use all these training instances when a new case is presented to be classified. This is unlike other eager learning models like neural networks where pre-existing training instances must be used to come up with the models in order for them to be effective.

In addition to this, there are 4Rs in a CBR problem solving session: retrieval, reuse, revise and review. (see Appendix D)

Retrieval: This is the step where we find cases that are the most similar to our new problem. Typically, this is done by having a vector of attributes/features defining each case and the cases that have the closest distance to our new problem based on the euclidean distance on their attribute vector relative to our new problem are being selected. This is also known as kNN (K-nearest neighbor). Naturally, we are not limited to just kNN and we will be looking into other classifiers as well.
Reuse: As the step clearly describes, the solution to the best candidate from the Retrieval step is being reused here for our new problem if that is possible. If not, we will move to the Revise step.
Revise: Since the candidate solution is not suitable for the target case, we adapt it to fit the target case and to do so, we will need to adopt knowledge intensive methods. If such actions are not possible, the system fails to find a suitable solution for our target case.
Retain: The solution for the new problem is evaluated and based on the outcome, the system will decide if it should be retained. Since it is not possible to keep all the solutions in the case library ad infinitum, the library should be maintained.

3. kNN Classifier (Minkowski distance)

A simple explanation of how a kNN classifier works can be found Padraig,C & Sarah,J.D. (2021)[2] where the retrieval of the nearest neighbors are determined by the weighted distance of each feature between the new data-point and all the data-points within the data-set. The general formulation for the distance (also known as Minkowski distance) is given as such:

$$MD_p(\mathbf{q}, \mathbf{x}_i) = \left(\sum_{f \in F} |\mathbf{q}_f - \mathbf{x}_{if}|^p \right)^{\frac{1}{p}}.$$

For p = 1, that is commonly known as Manhattan distance and for p = 2, that is Euclidean distance.

The determined class of the new data-point can take on various forms and the simplest of them is to take the majority class among all the candidate data-points and assign it to the new data-point. It is also possible to assign more weights to closer data-points using a weighted distance voting mechanism where each vote on the class of the new data-point is determined by the inverse of the data-point candidates' distance to the new data-point:

$$Vote(y_j) = \sum_{c=1}^{k} \frac{1}{d(\mathbf{q}, \mathbf{x}_c)^p} 1(y_j, y_c)$$

There are certain weaknesses when it comes to using this simple model. As Se-Hak.C.& Young-Woong.K. (2020, July 20) pointed out, many previously experienced cases must be retrieved in order to classify a data-point. This means there is a great deal of memory that must be used to store these cases. In addition, the technique tends to select the same number of datapoints and under the scenario where target cases should consider more neighbors while

[2] Padraig,C & Sarah,J.D. (2021), k-Nearest Neighbour Classifiers - A Tutorial, *https://dl.acm.org/doi/fullHtml/10.1145/3459665*

others should consider less. Lastly, there might be a scenario where there are too many neighbors that are equidistant to the targets and the deviation from desired similar neighbors might lead to lowering predictability.

4. Geometric Case Based Reasoning: Shape Distance Method for KNN

To tackle these problems, Se-Hak.C.& Young-Woong.K. (2020, July 20) proposed a shape distance method that selects nearest neighbors according to a slope similarity between two cases. Instead of simply computing the Euclidean distance between the two points, a consecutive time series is derived from the rise and drop of an observation of the current value versus the previous value of a data-point across time. (Imagine this to be the price of a stock)

To determine the shape distance between two cases, we use the following formulation:

$$d_i = n - \sum_{k=1}^{n} U(S_t * S_{t-k})$$

The dot product of the two time-series will yield a sign that the binary function U will return a value 1 if both values match. If all n samples match, you will notice that the distance is 0. However, if nothing matches, the function U returns 0 for all samples and that means the distance is n (the greatest).

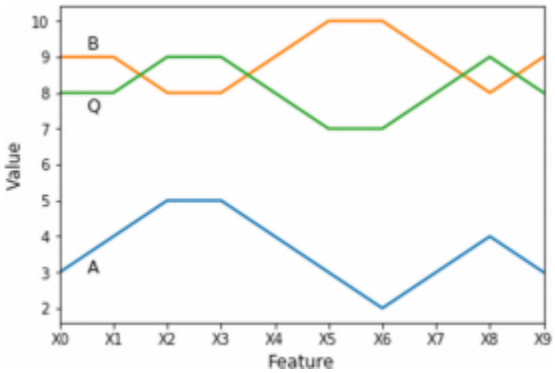
Please see Appendix E for the full algorithm.

5. Other similarity functions (Cosine Similarity and Correlation)

In this section, we look at other similarity functions before moving onto the different optimization methods: Cosine similarity and correlation.

Cosine similarity uses the angles between feature vectors rather than the distance to measure similarity. This means that while two data points (I.e. A and B) might be similar in distance, another data-point C can be closer if the angles C and one of the other points (A or B) is closer to each other.

Correlation, on the other hand, uses similarities in the standard deviation of value obtained from the feature space to derive similarity between a query and a data-point. An easier way to illustrate is in this diagram from Padraig,C & Sarah,J.D. (2021)'s paper.



While it is clear that B and Q are closer in distance, the patterns exhibited by A are closer to Q (increase-decrease-increase-decrease).

6. Kd Trees

As mentioned in the section 3 above, one of the weaknesses of the conventional kNN is the great deal of computation and memory that is required as a great deal of cases need to be retrieved. There are various speed-up mechanisms that can speed up this retrieval process and one of them is Kd trees. The idea behind Kd trees is simple which simply includes a binary tree to partition the dataset with training sets sorted to the leaves of the tree. This offers the potential for retrieval time that is O(d(log(n)) rather than O(dn), which is the complexity for a brute force search KNN. There is a weakness for this technique when the value of K (which is the number of neighbors required) is too large and the query time will take even longer than a brute force search. Another weakness of Kd trees is that it is still susceptible to the curse of dimensionality[3].

7. Approximate k-NN

Our next look into a more efficient mechanism is approximate k-NN. The idea here is that for most applications, it is not essential to retrieve the closest neighbors and we can adopt a 'good enough' approach. There are two popular strategies when it comes to this approach: Locality Sensitive Hashing and Random Projection Trees.

Locality Sensitive Hashing is an idea to map similar items into the same "buckets" with high probability. Using different variants of LSH, it allows us to come up with a candidate set to find the nearest neighbors that will be sub-optimal.

On the other hand, random projection trees is an extension of Kd trees and is based on the notion that there are two steps when it comes to the search for the nearest neighbor: the first step is to locate the right leaf node in the tree where our nearest neighbors are located and the second step is to backtrack in order to find better candidates or prune away sections of the tree so that we can avoid searching those sections.

The two methods used by random projection trees are defeatist search and multiple trees. In defeatist search, we simply search within the correct leaf node but the backtracking process is greatly curtailed so in a way, the search is given up in an early sense, giving the name defeatist. For the multiple trees method, the search is done in the correct leaf node without the backtracking process but on multiple variants of the tree itself. To create varieties in the tree variants themselves, it is common to randomly project the data into a different space.

Overall, Padraig,C & Sarah,J.D. (2021)'s study has shown that approximate k-NN is able to greatly improve retrieval times with little impact to the accuracy.

8. Artificial Neural Networks (ANN) in CBR

While we have looked into various types of k-NN techniques so far, we also looked into how ANNs can be integrated into CBR.

Xiaomeng,Y. (2019) introduced a three layer feed forward ANN (Chen, D., and Burrell, P. 2001) that replicates a CBR system without adaptation (see Appendix F). The input layers have one node for each problem feature while the hidden layers have one node for each stored case (I.e. datapoint). There are two types of links that link from the input nodes to the stored case node: weighted sum links that put emphasis on the importance of that feature and a product link if the

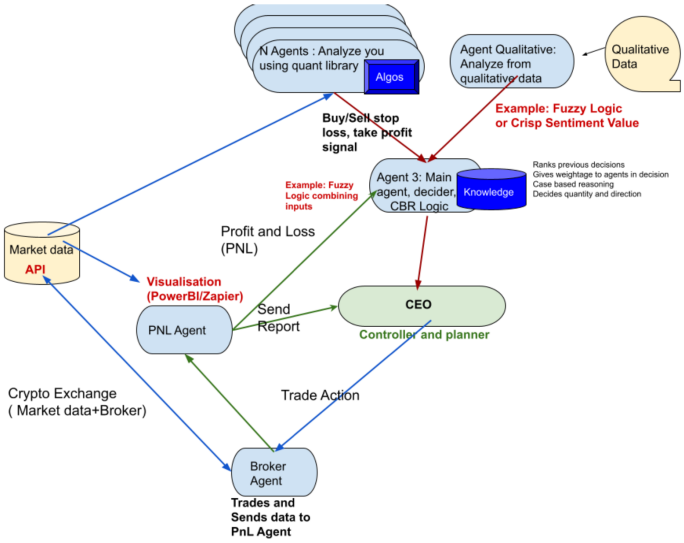
[3] Jon M. Kleinberg. (1997). Two algorithms for nearest-neighbor search in high dimensions. In 29th ACM Symposium on Theory of Computing. 599–608.

feature is essential for the activation of the case. Modification rules also allow the changing of the outputs coming out from the input nodes as well. The case nodes naturally link to the output nodes, which serves as the decision nodes. Each decision node represents a decision made from the retrieved cases and we can choose the decision in reusing the case with the highest activation.

9. Our Proposal (so far)

Our proposal for a fully-automated algorithmic trading solution lies heavily in a centralized model (I.e. MASST). There are naturally limitations that we need to consider in our own implementation (I.e. computing resources, development time, platform used, complexity of algorithms used, availability of training data-sets) and by no means, this proposal is the final design of our implementation. For instance, we might favor a k-NN implementation instead of a neural network later on in our implementation phase for our CBR retrieval phase.

Here is an overview of the architecture of our implementation:



In this section, we discuss the potential techniques that we are likely to use from the research that we have done so far above. We also discuss the pros and cons of the techniques that we have researched in the context of stock trading:

1. We have a CEO agent who serves as the central controller (which also serves as the profiling agent), which is similar to the coordinating agent that is discussed above. While this might serve as a single point of failure, we can ensure that the data and the state of the CEO agent will not get lost by saving its essential data to the database.

2. From the diagram above, it is quite clear that we will have a ‘prediction’ agent that seeks out solutions from previous cases (I.e. CBR). Stock chart pattern readings (E.g. head and shoulders) are very common and thus, there is no reason to omit this aspect.

3. We remove the fundamental analysis agent (FAA) from our architecture as our application focuses more on shorter positions and day trading. Moreover, our TAA contains N agents (each algorithm like SMA is running as a single agent), which serves as an ensemble with weights assigned to each of their output.

4. For Agent 3, we are likely to use an ANN for the retrieval of cases despite the fact that k-NNs are easier to implement (I.e. There are a lot of implementation samples online). This is a matter of scale as we are able to train an ANN beforehand without an ever-growing number of samples due to k-NNs lazy learning approach. As we are running the agents on the cloud, we do not have the luxury of a lot of memory at hand to hold these data-points. The training of an ANN can be done offline so there is no concern of the ANN in stalling during the retrieval process.
   Agent 3 also serves as a risk management agent where it decides the quantity to trade and the direction based on the outcome of CBR.

5. We also have a broker agent that automates the trades whilst the research papers that we see merely sends the notifications to the user interface for further action.

6. This also means that we will need to provide a kill switch to hard brake on the system if anything goes wrong. That is provided by the CEO agent.

7. We also provide a qualitative agent that analyzes news, tweets and posts on social media that might affect the price of the asset or market sentiment. This is not proposed in any of the papers.

10. Future Direction

As mentioned earlier, the architecture design and the design decisions that we have made are, by no means, final and there are certainly other research directions that we would like to take to improve the effectiveness of the model that we have. Here are some of them:

1. An hybrid approach of using ANNs and k-NNs: The use of ANNs to train the weights of the k-NN method is something that can be considered.

2. ANNs can also be used for training the weights of the ensemble as well.

3. Integrating ANNs during the reuse/revise phase of CBR. This has not been discussed much in this paper but the adaptation of the selected case to the query might not always be easy. Xiaomeng,Y. (2019) talks about the integration of ANNs in the adaptation model of CBR to address this issue.

11. Conclusion

This study discusses the different types/variants of MAS and their components briefly and we look into CBR with a focus on the retrieval phase where we visited various methods like k-NNs, its variants and ANNs. We also briefly discussed our proposal, the choices we made based on the research material we have and the pros and cons of the methods in consideration of the context of trading.

References

Luo, Y. & Davis, Darryl N. & Liu, Kecheng. (2008). A multi-agent framework for stock trading.

Renuka,K., Priyanka,P. & Yogendra,K. (2018). A State of Art Review on Various Aspects of Multi-Agent System, *Journal of Circuits, Systems, and Computers*

Elhadi, S & Sadd, A. (2005) MULTI-AGENT SYSTEM ARCHITECTURE TO TRADING SYSTEMS, *Journal of Interconnection Networks*

Tirea, M.,Tandau,I. & Negru,V.(2011). Multi-agent Stock Trading Algorithm Model, 2011 IEEE

Giorgio, L., Luigi,P., Paolo A. & Marco,V. (2016) A Smart Financial Advisory System exploiting Case-Based Reasoning, *FINREC 2016*

Se-Hak.C.& Young-Woong.K. (2020) Geometric Case Based Reasoning for Stock Market Prediction

Padraig,C & Sarah,J.D. (2021), k-Nearest Neighbour Classifiers - A Tutorial, *https://dl.acm.org/doi/fullHtml/10.1145/3459665*

Jon M. Kleinberg. (1997). Two algorithms for nearest-neighbor search in high dimensions. *In 29th ACM Symposium on Theory of Computing.* 599–608.

Xiaomeng,Y. (2019). The Integration of Artificial Neural Network in Case-based Reasoning

Chen, D., and Burrell, P. (2001). Case-based reasoning system and artificial neural networks: A review. Neural Computing & Applications 10(3):264–276.

Appendix A

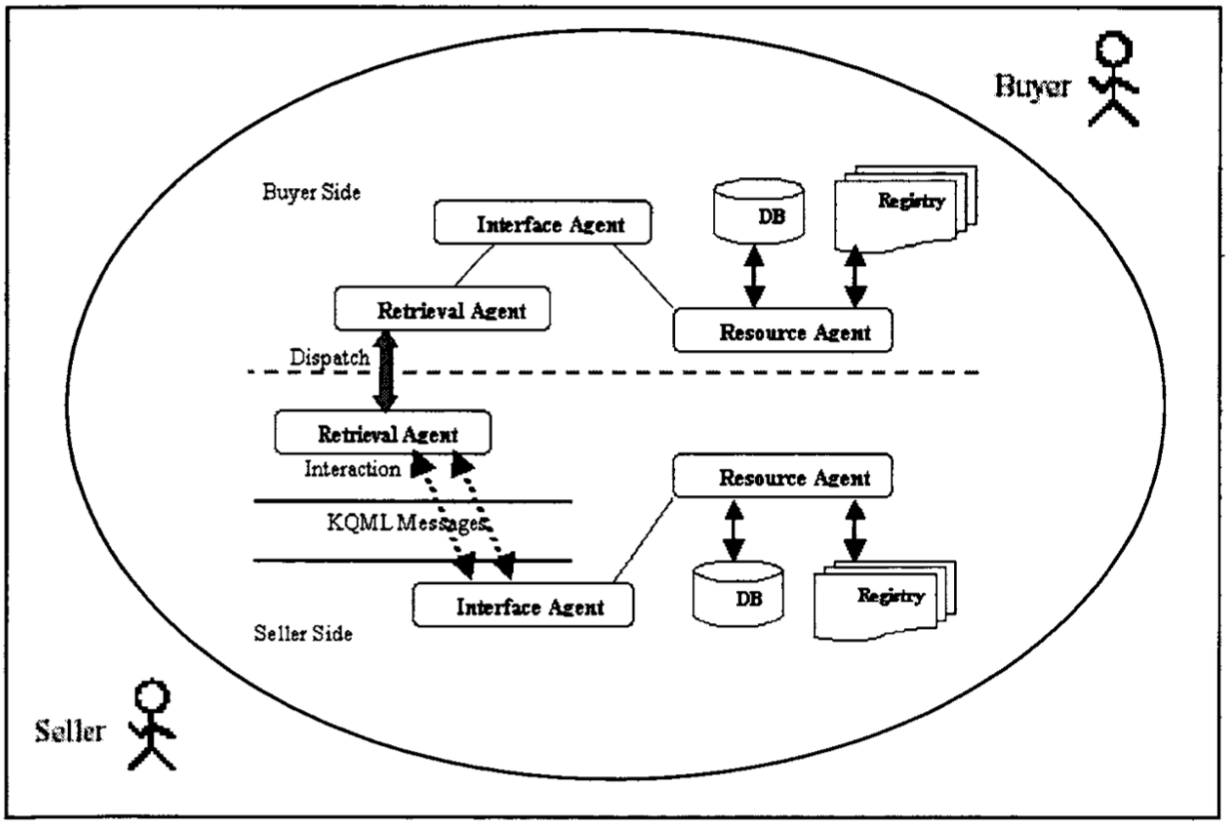


Fig. 1. PPMASOT system architecture.

Appendix B

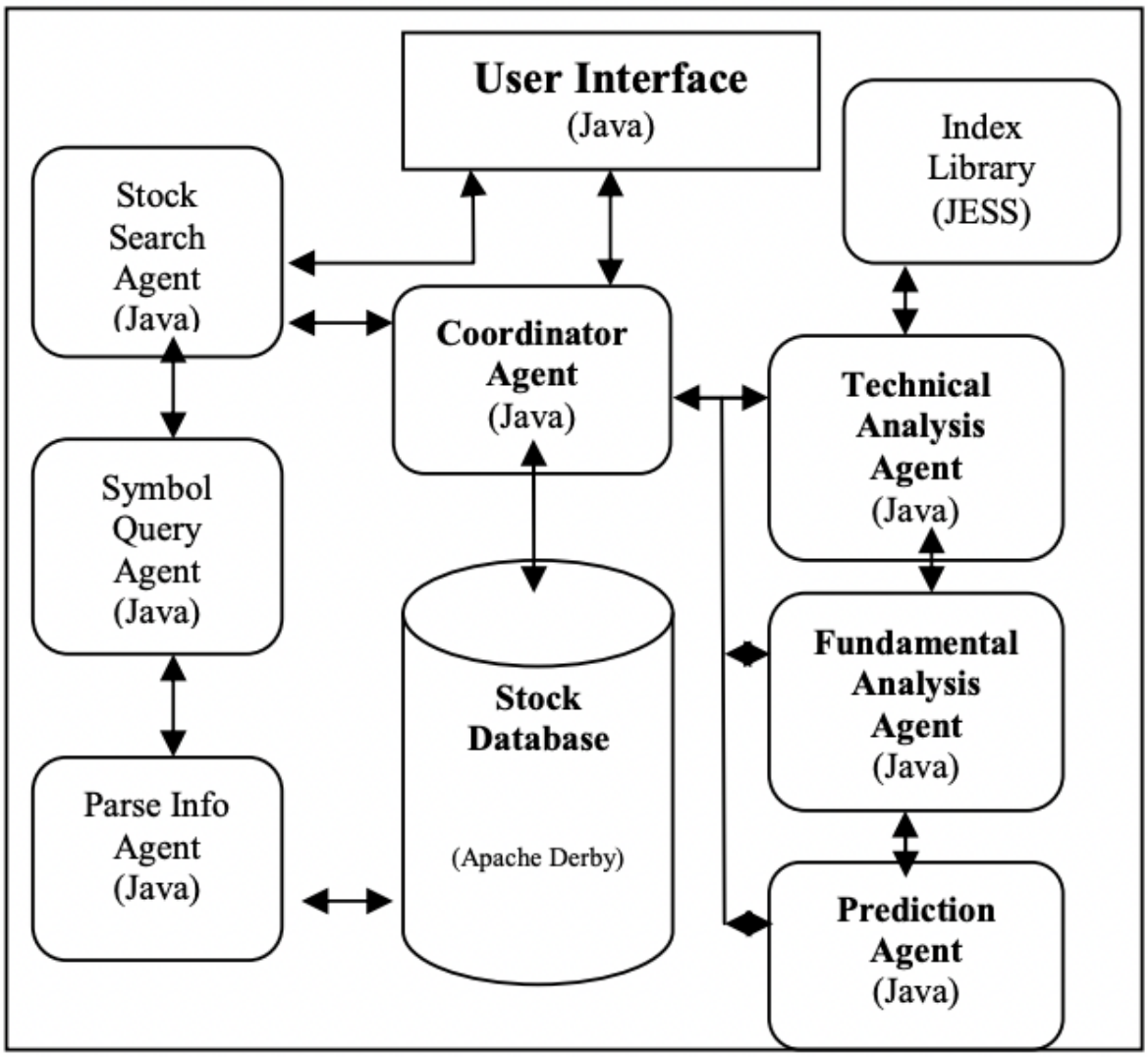


Figure 1. Multi-Agent System Architecture

Appendix C

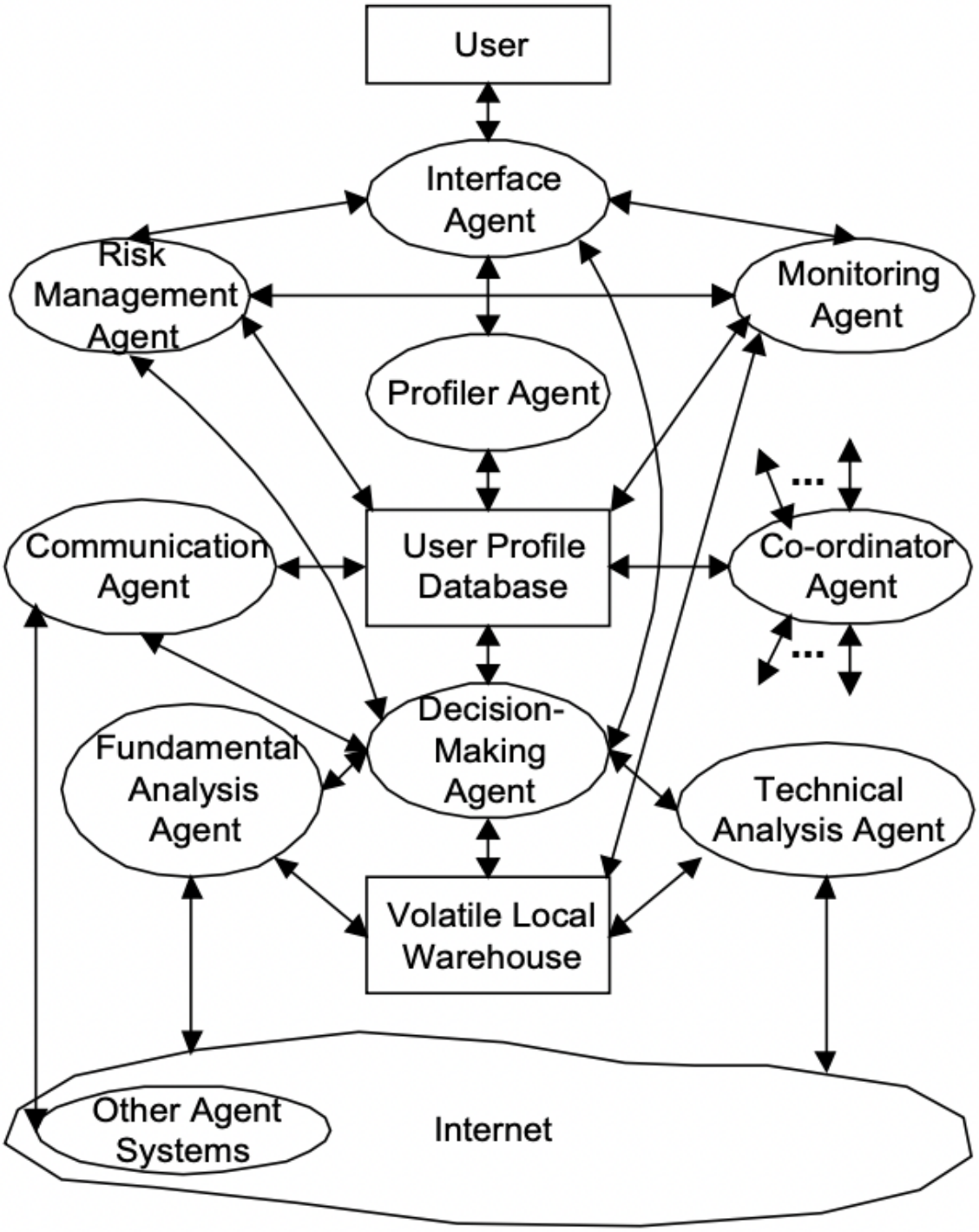


Figure 1. MASST Framework

Appendix D

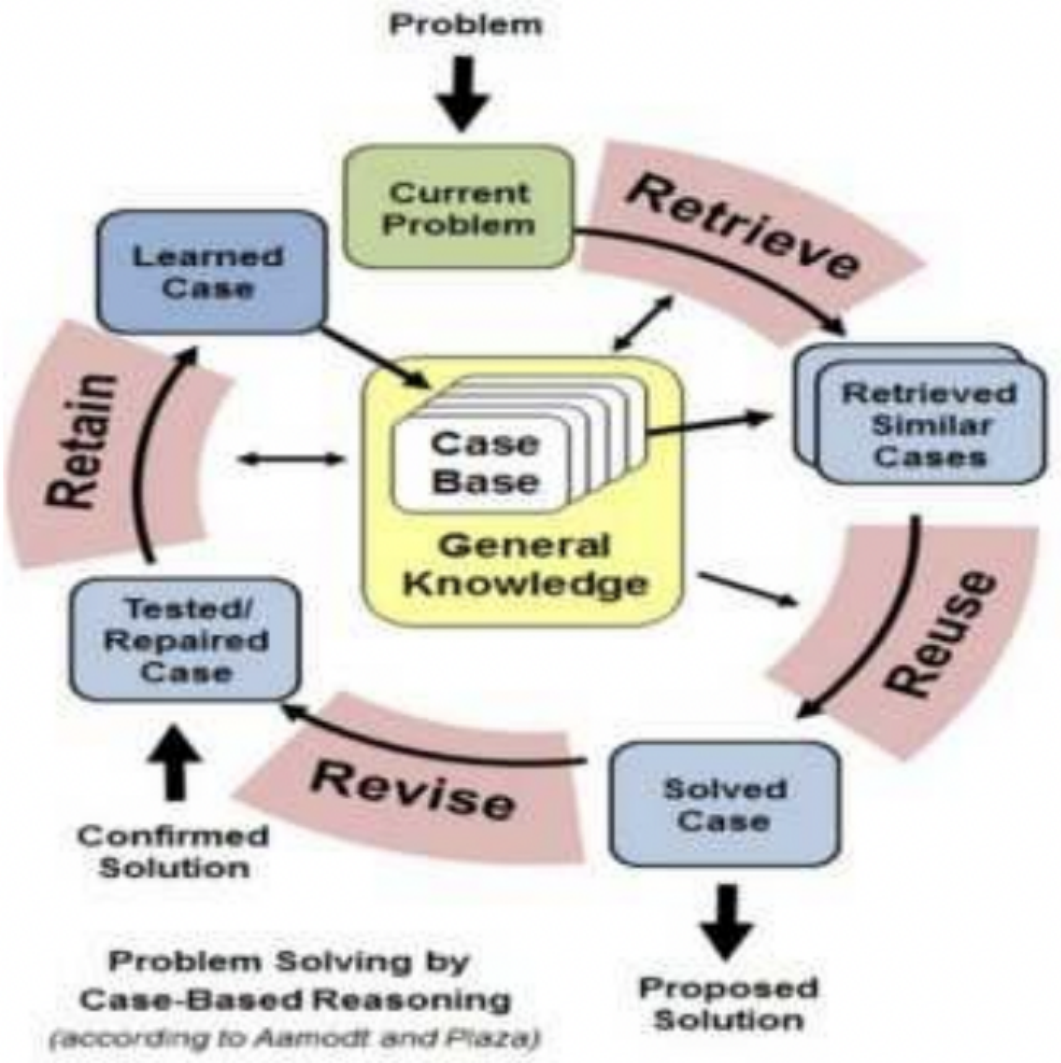


Figure 2. The 4R's CBR cycle

Appendix E

1. Perform exploratory data analysis (EDA): identify overall patterns and outliers.
2. Transform data for comparability (for example, differencing).
3. Perform a CBR machine.
   a. Obtain a shape information (fall or rise) from comparing a target case (t) with previous time series data (t-1).
   b. Seek the j neighboring cases x_k in the past which are closest to x_t according to the distance function: $d_i \equiv n - \sum_{k=1}^{n} \cup (S_t * S_{t-k})$.
   c. Compute the sum of weights: $d_T = \sum_{s=1}^{j} d_s$.
   d. Determine the relative weight of i^{th} neighbor: $w_i = \frac{1}{j-1}\left(1 - \frac{d_i}{d_T}\right)$.
   e. Find the successor x_{k+1} of each case x_k in the set of neighbors.
   f. Calculate the forecast for x_{t+1} as the weighted sum of successors: $\hat{x}(t+1) = \sum_{i=1}^{j} w_i x(t_i + 1)$.
   g. Do loop from a to f until a final period of prediction in the test phase.
4. End Case Based Reasoning.
   → Calculate result from the model.
5. Detransform the variables.
   → Analyze results.

Appendix F

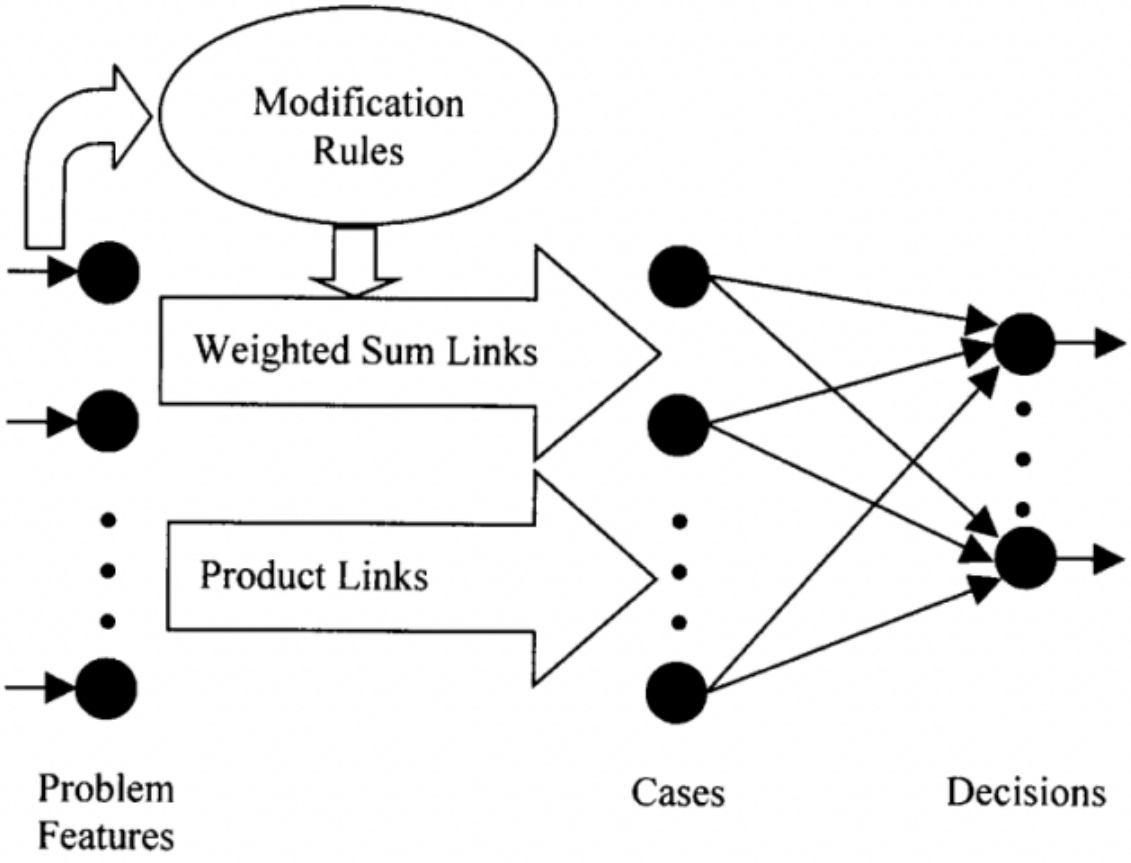


Figure 1: An ANN Model for CBR(Chen and Burrell 2001)