Failure-Driven Transformational Case Reuse of Explanation Strategies in CloodCBR*

Ikechukwu Nkisi-Orji[1], Chamath Palihawadana[1], Nirmalie Wiratunga[1], Anjana Wijekoon[1], and David Corsar[1]

School of Computing, Robert Gordon University, Aberdeen, UK
{i.nkisi-orji,c.palihawadana,n.wiratunga,a.wijekoon1,d.corsar1}@rgu.ac.uk

Abstract. In this paper, we propose a novel approach to improve problem-solving efficiency through the reuse of case solutions. Specifically, we introduce the concept of failure-driven transformational case reuse of explanation strategies, which involves transforming suboptimal solutions using relevant components from nearest neighbours in sparse case bases. To represent these explanation strategies, we use behaviour trees and demonstrate their usefulness in solving similar problems. Our approach uses failures as a starting point for generating new solutions, analysing the causes and contributing factors to the failure. From this analysis, new solutions are generated through a nearest neighbour-based transformation of previous solutions, resulting in solutions that address the failure. We compare different approaches for reusing solutions of the nearest neighbours and empirically evaluate whether the transformed solutions meet the required explanation intents. Our proposed approach has the potential to significantly improve problem-solving efficiency in sparse case bases with complex case solutions.

Keywords: case-based reasoning, case reuse, explainable AI, behaviour trees

1 Introduction

iSee project uses case-based reasoning (CBR) to systematically capture, retrieve, reuse, revise, and retain user explanation experiences for the benefit of other users. The primary objective is to establish a comprehensive repository of best practices, which can subsequently pave the way to compliance in explainable AI (XAI). An explanatory experience, in its most fundamental form, comprises two key components: the explanation requirement (a detailed description of the case problem) and the suggested explanation approach (a thorough depiction of the case solution).

The problem description should primarily capture the enquiries for which end-users are expected to seek answers through an explanation. However, the

* This research is funded by the iSee project. iSee is an EU CHIST-ERA project which received funding for the UK from EPSRC under grant number EP/V061755/1.

problem description encompasses more than just these queries. It must also incorporate the context, considering factors such as the application domain, the black-box model, the data utilised in model creation, and crucially, the end-users' knowledge levels. By addressing these aspects, an explanation experience within the iSee project guarantees that the recommended strategies cover all facets of XAI. The solution description captures the explanation strategy, which unlike a single explanation captures alternative ways in which end-users' explanation needs can evolve. Recent work suggests that multiple explanations (such as factual, semi-factual, and counterfactual) using alternative explanation techniques are necessary to achieve user satisfaction and engagement with XAI [11, 12]. In iSee, an explanation strategy is captured as a behaviour tree [14] in order to manage transitions between alternative explanation styles in an interactive manner (e.g., conversational interaction using a chatbot).

Given the complex structure (as opposed to a class label) and inherent sparseness of the collection of explanation experience cases, it is unlikely that a solution derived from a single neighbouring case would be sufficient to satisfy the end-user's explanation needs. This circumstance necessitates the implementation of efficient reuse operations to ensure that the CBR system can combine explanation strategies from multiple cases. In particular, we examine the impact of failures in aligning with end-user questions on suboptimal explanation strategies, and alternatively, identify neighbouring cases wherein entire or partial explanation strategies can be amalgamated based on varying degrees of correspondence. Specifically, we study the impact of failures in matching end-user questions which lead to suboptimal explanation strategies and instead identify neighbouring cases wherein entire or partial explanation strategies can be combined based on levels of match. Since the underlying CBR engine used with iSee is CloodCBR [9], it becomes important to implement reuse operators that are compatible with the microservices architecture upon which CloodCBR is built. Accordingly, the key contributions of this paper are as follows.

- Formalise a novel reuse operator as a transformational adaptation method for CloodCBR aligned with the modular and scalable design requirements of its microservices architecture.
- Evaluate the effectiveness of the reuse operator and provide insights into its operation.

The remainder of this paper is structured as follows. Section 2 provides a brief overview of previous work on adaptation techniques for case reuse. Section 3 introduces our failure-driven reuse technique and the updates to CloodCBR to support the reuse operation. Section 4 measures the extent to which the reuse operation satisfies the intent of the query and discusses our findings. Section 5 concludes the paper with considerations for future work.

2 Related Work

There are two main approaches to learning adaptation knowledge for case reuse: weighted majority voting and case difference heuristic (CDH) [15]. Weighted

majority voting is a method used in k-NN retrieval to determine the solution of a query by calculating the weighted majority of the solutions of the k-nearest neighbours [6, 13]. To determine the optimal value of k, several techniques in this method employ a leave-one-out test that computes a set of n closest neighbours for each query case, where n ranges from 1 to the size of the case base. Weighted majority voting is applied to each set of neighbours to calculate the solution of the query, which is then compared to the actual solution. The value of n that results in the best overall quality of the CBR system is used as the value of k in the future. The CDH approach, first proposed by Hanney and Keane [3], involves pairing cases from the case base and using each pair to learn a rule for adapting one case to another [4, 5, 7, 16]. Liao, Liu, and Chao [5] proposed a CDH adaptation method for regression tasks that employs a case difference heuristic approach and a neural network to learn the difference characterisation. Their approach involves training a network to map problem differences to differences in output values, eliminating the need for pre-defined generalisation strategies. Additionally, Jalali, Leake, and Forouzandehmehr [4] used the CDH approach for classification by using a statistical method to generate case adaptation rules while Ye, Xiaomeng, et al. [16] introduced an approach that uses neural networks to learn adaptation knowledge from pairs of cases. Unlike the case solutions in this paper, these adaptation techniques work with simple structures such as class labels in classification tasks and numbers in regression tasks.

3 Failure-Driven Reuse

Organising multiple end-user enquiries into distinct intents can streamline their management and improve efficiency. This approach involves systematically categorising questions based on their underlying purpose or intent, resulting in more coherent and targeted responses to each group of enquiries. Additionally, this organisation facilitates the evaluation of failure for Reuse by identifying the extent to which questions have been matched during the retrieval process and the degree to which intents have been addressed.

3.1 Case representation

In iSee, a case c consists of attributes which include a user intent and a set of questions that express that intent as shown in Figure 1. The explanation strategy for the intent-related questions forms the case solution. Other attributes in the case description are used to express the context of the explanation strategy such as the attributes of the AI Model (e.g., AI task and dataset type), the explanation criteria (e.g., explanation scope) and the user/user group (e.g., knowledge level). The attributes that form the representation of the case are formalised by iSeeOnto [1].

[1] iSee ontologies that form part of the iSee project https://w3id.org/iSeeOnto/explanationexperience

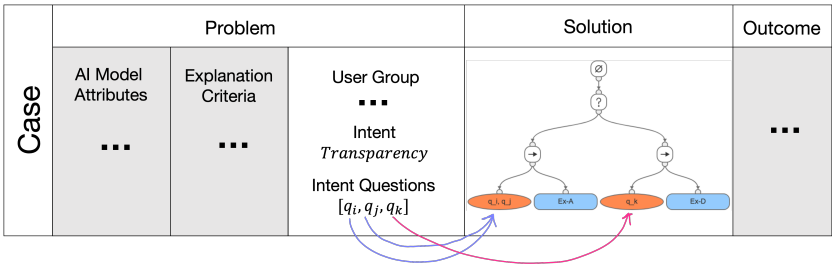

Fig. 1. Explanation Experience Case structure

As depicted in Figure 1, the focus of this paper is the explanation intent that is expressed as questions and how the solution aligns with the questions to satisfy the intent. For example, a user can express their explanation need, i.e. intent for transparency of the AI model or the decisions of the AI model by using one or more questions. A complete case should consist of a solution that addresses each question to the user's satisfaction.

Case solution is an explanation strategy modelled using Behaviour Trees (BT). A BT is a conceptual model that formalises the behaviours and navigation of an entity in an environment [1]. We adapt BTs for designing explanation strategies by introducing behaviours such as detecting an intent or a question and executing an explainer. Table 1 presents the nodes and functionalities used in the Explanation Strategy BT design.

Type	Node	Description
Composite	Sequence	Has one or more children nodes and the children nodes are executed from left to right until one fails.
	Priority	Has one or more children nodes and the children nodes are executed from left to right until one succeeds.
Condition	Intent	Given user intent, checks if it matches the node intent.
	Question	Given user question, checks if it matches the node question.
Action	Explainer	Given context of an explanation requirement including data instance and AI model decision, execute the explanation technique to generate an explanation.

Table 1. Explanation Strategy Behaviour Tree Nodes

A sub-tree that consists of a Sequence Node with a Question Node and an Explainer Node as children is considered a self-contained sub-tree that is not affected by (or influences) other parts of the tree. Accordingly, an explanation strategy can be seen as a collection of such sub-trees that essentially guides the user to receive explanations as they raise questions. Figure 2 presents an example explanation strategy which can be interpreted as follows. If the intent

is transparency and the user expressed this intent by posing *Q-A* question, the Integrated Gradients explainer is executed to provide an explanation. If the user has a follow-on question *Q-B*, the Nearest Neighbour explainer is executed. Similarly, if the user asked a Q-C question in relation to the performance intent, the AI Model Performance explainer is executed. Note that questions A, B and C can be pre-set questions which an AI system designer have considered to be relevant to end-users who might be seeking explanations. For instance, in a loan applicant assessment system aimed at determining eligibility for a loan, the question "how can I get a different outcome" represents an anticipated inquiry from applicants and falls under the category of actionability intent. Figure 3 presents examples of questions classified by their corresponding intents, as used in this work. The case solution BT is also formalised by iSeeOnto.

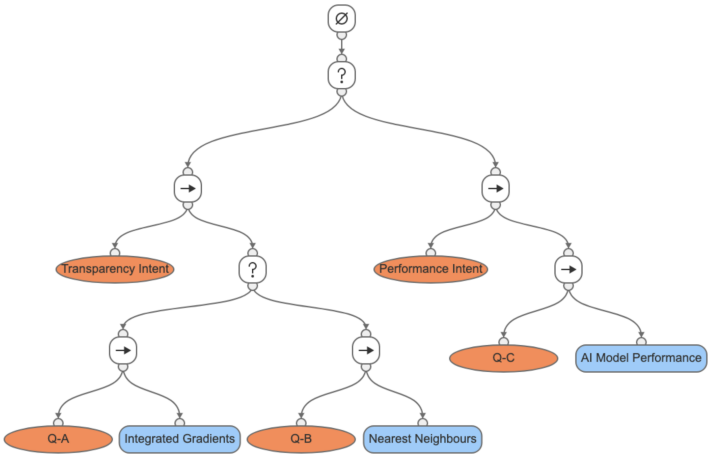

Fig. 2. A case solution modelled as a Behaviour Tree

3.2 Reuse failure and Transformation using Gale-Shapley Algorithm

Due to the complex structure of BT case solutions and the sparse collection of explanation strategies, a single neighbouring case may not provide sufficient explanation for the end-user needs. Thus, we propose a reuse operation to combine explanation strategies from multiple cases. Specifically, we align end-user questions with case questions using the Gale-Shapley algorithm [2] to amalgamate entire or partial explanation strategies from neighbouring cases.

The Gale-Shapley algorithm, also known as the deferred acceptance algorithm, is a mechanism for solving the stable marriage problem. The problem involves finding a stable matching between two sets of equal size, such as men and women, doctors and hospitals, or students and schools. The algorithm works by having each member of one set, say the men, propose to their most preferred

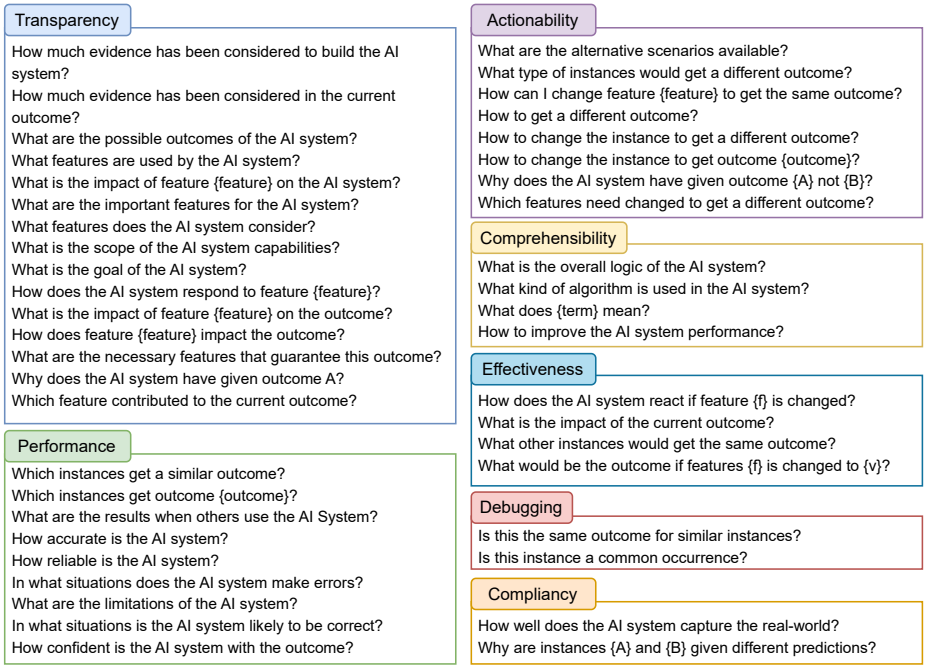

Fig. 3. Examples of question templates for different explanation intents

member of the other set, say the women. Each woman then reviews her proposals and rejects all but her most preferred suitor. The men who were rejected then propose to their next most preferred woman, and the process repeats until every woman is engaged. The algorithm guarantees that a stable matching will be reached, meaning that there are no two pairs who would both prefer to be with each other instead of their current partners. In addition, the matching is "men-optimal", which means that every man is matched with his most preferred woman among all possible stable matchings, and "women-pessimal", which means that every woman is matched with her least preferred man among all possible stable matchings.

Given a query, we perform a retrieval of K nearest cases using an appropriate retrieval strategy. Starting with the closest case to the query, we match the query questions with the case questions using the Gale-Shapley algorithm. The preference order of a question in one set is determined by the similarity of that question with every question in the other set. We use Sentence-BERT (SBERT), a variant of the pretrained BERT network that utilises siamese and triplet network architectures to derive semantically meaningful sentence embeddings [10]. When applied to the question texts, the resulting embeddings are then compared using cosine similarity. After a stable matching is found, we determine the average similarity of the returned question pairs using SBERT and check if the score is up to an acceptance threshold, α. If the average similarity score reaches the

threshold, we stop and reuse the explanation strategies that are associated with the matched case questions. Otherwise, we increase the case neighbourhood by 1 and apply the Gale-Shapley algorithm again. This process of increasing the neighbours continues until either the average score of the question pairs reaches α or we have considered all K cases.

The matching process is as shown in the recursion in Equation 1. NN_i is the set of questions in the i^{th} nearest case retrieved and K is the maximum number of cases that are retrieved for the reuse operation. We initialise α to a real value in the interval $[0, 1]$, $i = 1$ and $L_q = NN_i$.

$$MATCH(c_q, L_q) = \begin{cases} \text{pairs, score } = match(c_q, L_q), & \text{if score } \geq \alpha \text{ or } \ i = K \\ MATCH(c_q, L_q + NN_{i+1}), & \text{otherwise} \end{cases} \tag{1}$$

The function $match(c_q, L)$ in Equation 1 implements the Gale-Shapley algorithm and is executed whenever we call $MATCH(c_q, L_q)$. Operation $L_q + NN_{i+1}$ increases i by 1 and also increases the neighbourhood for consideration of reuse by 1. In our experiments, we observed that α values in the range of 0.6 to 0.9 yielded positive results. $match(c_q, L)$ matches unequal sets (that is, query questions and questions of retrieved cases) as follows.

1. Let c_q be the set of query-associated questions and L_q be the set of questions from k nearest cases that are retrieved for the query
2. For each $x \in c_q$, create a preference list $pref(x, L_q)$ that orders the elements of L_q by similarity (decreasing) to x. Create similar preference lists for each $y \in L_q$ for the elements of c_q
3. Initialize all $x \in c_q$ and $y \in L_q$ to be without partners (unpaired)
4. While there exists at least one x without a partners and $y \in L_q$ which x has not attempted to pair with:
   (a) Choose x
   (b) Let y be the highest-ranked L_q in $pref(x, L_q)$ to whom x has not yet attempted to pair with
   (c) If y is free (i.e. not already paired), then pair x and y
   (d) Otherwise, if y is currently paired to another $\overline{x} \in c_q$, then compare x and $\overline{x}$ using y's preference list ($pref(y, c_q)$):
      i. If y prefers x over $\overline{x}$, then break the pairing between y and $\overline{x}$ and pair x and y
      ii. Otherwise, x remains unpaired and continues the attempt to pair with the remaining $\overline{y} \in L_q$
5. The algorithm terminates when every free c_q has attempted to pair with every member of L_q.

When the matching method is terminated, we return *pairs* (pairs of matched query and case questions) and *score* (average similarity of *pairs*). The solution that is being constructed from *pairs* is considered to have failed whenever *score* is less than α.

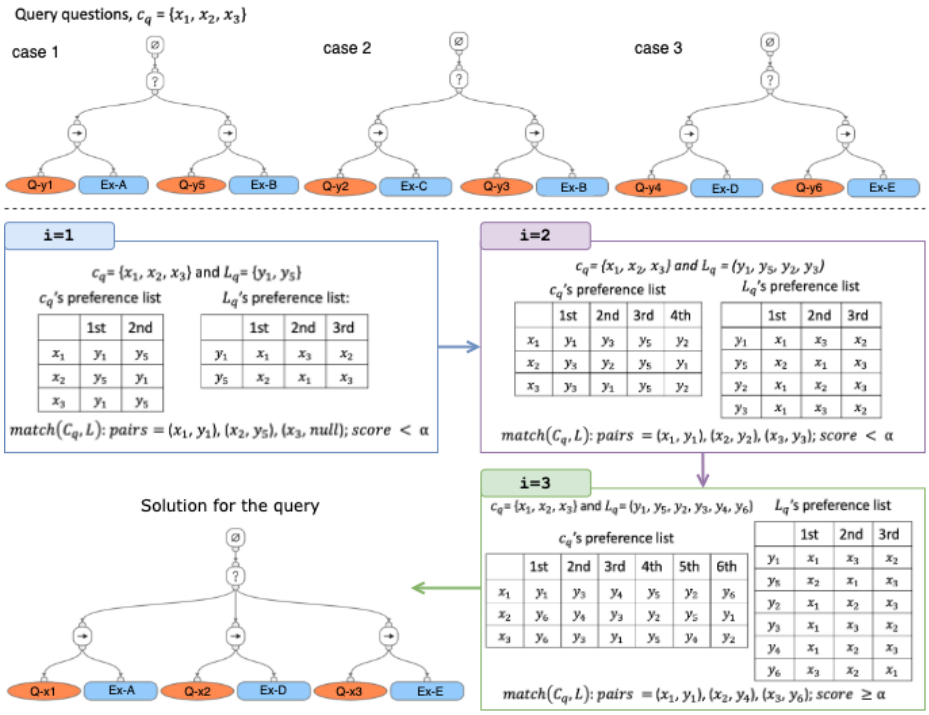

Fig. 4. Example showing how the solution of a query case is transformed using the nearest cases

We demonstrate how $MATCH(c_q, L_q)$ works using the example in Figure 4. The example shows the query case c_q with three questions and the solution parts of three closest cases that have been retrieved from the case base. In the first matching attempt (that is, $i = 1$) and L_q containing *case 1* only, the Gale-Shapley algorithm ($match(c_q, L_q)$) is used to determine a stable matching between c_q and L_q based on their preference lists. The list of preferences for c_q shows that x_1 prefers y_1 over y_2 because $sim(x_1, y_1) > sim(x_1, y_2)$. The similarity function $sim(x, y)$ determines the similarity of x and y and in our case, we used the cosine similarity of vector representations from SBERT. Using the similarity function, the preference list for y_1 shows that $sim(y_1, x_1) > sim(y_1, x_3) > sim(y_1, x_2)$. With the preference lists, x_1 matches y_1 to form the first pair of $match(c_q, L_q)$ and x_2 matches y_5 as the second pair. When we attempt to match x_3, we find that its first preference y_1 is already paired with x_1. We consult the list of preferences of y_1 which shows that y_1 prefers x_1 over x_3. Consequently, the pair (x_1, y_1) remains unchanged. Continuing the attempt to pair x_3, we check the next entry in its preference list, which is y_5. As y_5 is already paired with x_2 and y_5 prefers x_2 over x_3, x_3 remains unpaired in this round. At this point, we

determine *score* of the pairs as:

$$score = \frac{sim(x_1, y_1) + sim(x_2, y_5)}{|c_q| = 3}.$$

Assuming *score* is below the acceptance threshold α, we increase i which expands L_q to include the next closest neighbour. $match(c_q, L_q)$ is repeated which can undo previously matched pairs. For example, at $i = 2$ in Figure 4, y_2 was found to be a better match for x_2 than its previous pairing. This results in an update of the pair from (x_2, y_5) to (x_2, y_2). When the algorithm ends (that is, when $score \geq \alpha$ or all retrieved cases have been considered), we construct the BT solution (explanation strategy) with the final pairs using the relevant parts of the matched cases.

3.3 Solution construction and execution

A constructively adapted explanation strategy is executed as part of a conversational interaction, which is also modelled as a BT. While the complete interaction model is not within the scope of this paper, Figure 5 shows how an explanation strategy is incorporated into the interaction model. The interaction model is designed to be generalisable to any explanation strategy. Accordingly, at runtime, the explanation strategy dynamically replaces the Explanation Strategy placeholder Node. The immediate parent sub-tree is a "Repeat-until-success"

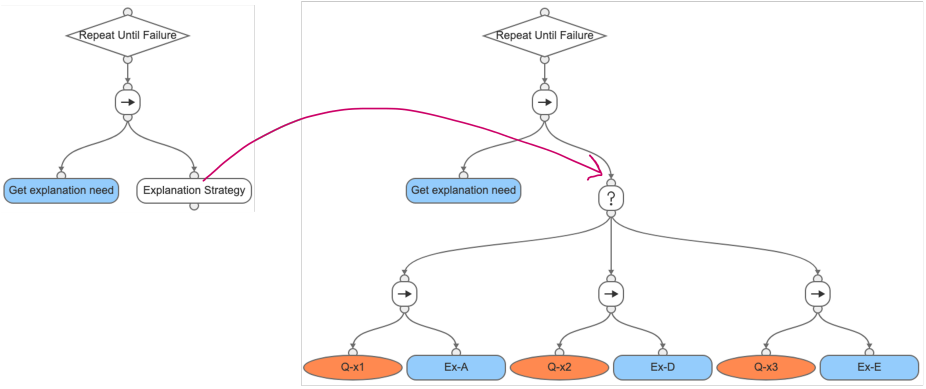

Fig. 5. Dynamic adaptation of the interaction model to execute the Explanation Strategy

node that iteratively interacts with the user to get their intent question (Get explanation need Node) and executes the respective sub-tree in the explanation strategy. This is repeated until a failure occurs. There are two cases which we consider as failures to exit the iteration: when the user indicates that they have no other questions, or when the explanation strategy is not able to answer user

questions. The latter is recorded as feedback for future improvements to the explanation strategy.

3.4 CloodCBR enhancements for the reuse operation

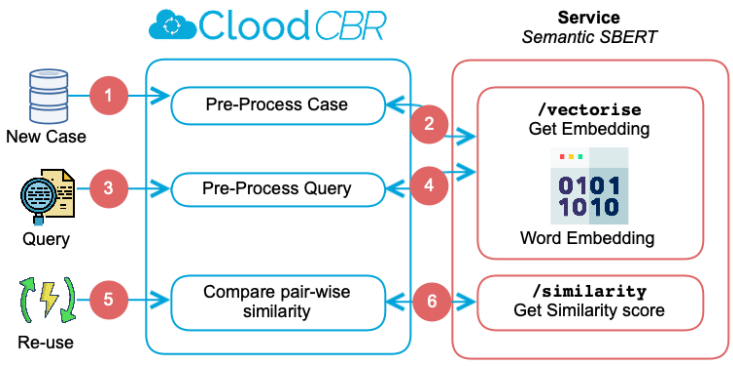

Fig. 6. CloodCBR Changes to Support Sentence Transformers

As CloodCBR is the CBR framework for iSee, we briefly discuss the changes to CloodCBR to support the reuse operation. CloodCBR's microservices architecture and extensibility facilitates the seamless integration of new similarity metrics. To incorporate state-of-the-art text embedding models (such as SBERT [10]), we first developed an independent *semantic-sim* service with API endpoints */vectorise* and */similarity*. The *vectorise* endpoint returns the text-embedding for a given string while the *similarity* endpoint vectorises and returns the cosine similarity between the embeddings of a pair of texts. The service was then integrated in CloodCBR as follows.

1. Case Creation - When a new case is added to the case base, CloodCBR undergoes a pre-processing step that can include various similarity metrics, such as ontology retrieval or text embedding. For attributes that have been configured to use Sentence-BERT for word embedding similarity, the word embeddings are generated via an API call to the semantic-sim service.
2. Query/Retrieval - When searching the case base for similar cases, the query case undergoes pre-processing according to the chosen similarity metrics. For an attribute that requires word embedding similarity, a request is sent to semantic-sim service to retrieve the word embedding. The vectorised query attribute value is then compared with the corresponding case embeddings using a cosine similarity script in the case base [8].
3. Post-retrieval/Reuse - When we need to compare texts using word embedding after case retrieval, e.g., determining the similarity of a pair of question texts in the Gale-Shapley implementation for reuse, we use a call to *similarity*.

The reuse operation is implemented as a Python function and forms part of the core service for managing the phases of the CBR cycle. We have open-sourced the integration and API service for semantic-sim on GitHub[2].

4 Evaluation

We conducted an experiment to evaluate the effectiveness of the reuse operation with CloodCBR by measuring the alignment between the intent of the constructed solution and the intent of the query.

4.1 Experiment setup

In the experiment, we generate cases using a bank of intents with associated questions as shown in Figure 3. We randomly assign a name and an intent to each case. Using the intent-question bank, we randomly sample questions for the selected intent based a Poisson distribution (lambda = 2). As shown in Figure 7, there are instances where no questions are sampled because the Poisson value is 0. In such situation, we repeatedly generate a new Poisson value until it is greater than 0. We ensure that every case in the case base is unique and that there is one intent per case for simplicity.

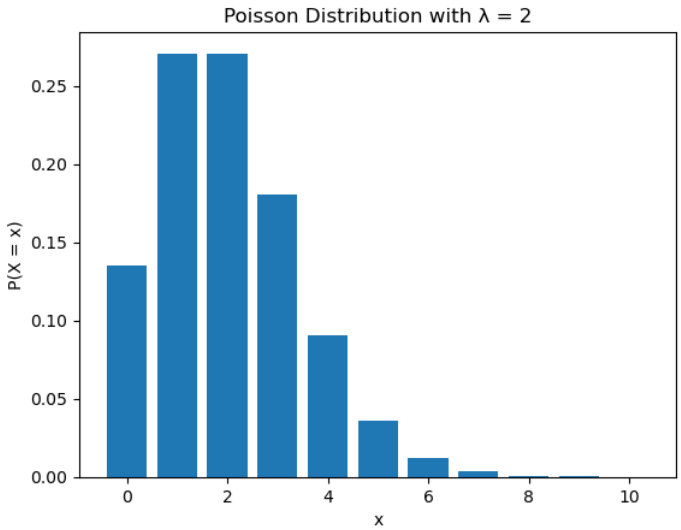

Fig. 7. Poisson distribution for sampling intent questions for the generation of cases and queries

[2] https://github.com/RGU-Computing/clood/tree/master/other-services/semantic-sim

Using the CloodCBR framework, we create and configure a CBR application with the case-base structure as described in Section 3.1. We investigate the reuse of cases for different case base sizes (20, 30, 40, 60, 80, and 100) and acceptance thresholds (0.5 to 0.9) using the introduced Gale-Shapley matching technique (GSA). We analyse the impact of deferred acceptance between iterations of $MATCH(c_q, L_q)$ (see Equation 1) by including a variant with non-deferred acceptance (NDA). In the NDA approach, any question-pair similarity that reaches α is not unpaired when L_q is increased in subsequent iterations. Reuse of closest-neighbour best match (BM) forms the baseline approach. Both GSA and NDA approaches can be viewed as transformations of BM by reusing solutions from multiple cases.

The queries are generated in a manner similar to the cases but have no solutions. We used a set of 50 queries for the experiments and retrieved the five nearest neighbours for each query ($K = 5$). The case solutions are the explanation strategies that are associated with the questions.

In the evaluation, we estimate the level of satisfaction with the intent of the query by assessing how well the intent of the constructed solution aligns with the intent of the query. Specifically, we analyse the extent to which the intent of the questions of the cases that were matched with the query aligns with the intent of the query. Recall that the algorithm of the GSA reuse operation returns a set of question pairs, P with each pair $< x, y >$ consisting of $x \in c_q$ and $y \in L_q$ as described in Section 3.2. Let I_x represent the intent of c_q and I_y represent the intent of the case to which y belongs. We define an intent satisfaction metric as a real number between 0 and 1 as shown in Equation 2.

$$\text{Intent Satisfaction} = \frac{\sum_{i=1}^{|P|}(I_q = I_{c_i})}{|c_q|} \tag{2}$$

Intent Satisfaction = 0 shows a complete mismatch between the intent of the question and the intent of the cases that were used to construction a proposed explanation strategy. At the other extreme, Intent Satisfaction = 1 show a complete alignment between the query intent and the reused cases.

4.2 Results and Discussion

The evaluation results in Figure 8 show the degree of satisfaction with the intent of the query after the reuse operation. The performance of BM, which always reuses the solution of the best-matched case, is represented as a line because it is not affected by α.

As expected, it was better to reuse the solutions of more cases than the best-matched case alone. GSA tends to outperform NDA at low α value (0.5). This is also expected since GSA uses the average similarity of pairings to check for failure, while NDA identifies failure per question. The impact is that NDA reuses more cases than GSA for constructing a solution. At $\alpha = 0.5$, GSA reuses 2 closest-neighbours cases, while NDA reuses 2.5 cases. GSA has the better

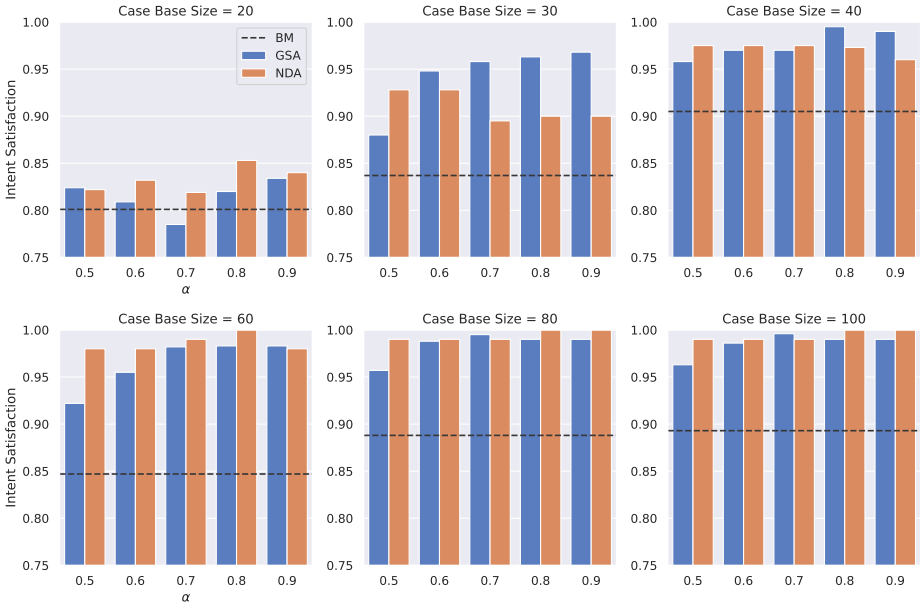

Fig. 8. Results showing the level of satisfaction with the query intent using different reuse methods for different case base sizes and acceptance thresholds (α).

performance when the case base is sparse (i.e., 30 and 40) while NDA was better for case base sizes 60 and 80. When the case base is relatively large (i.e. 100)

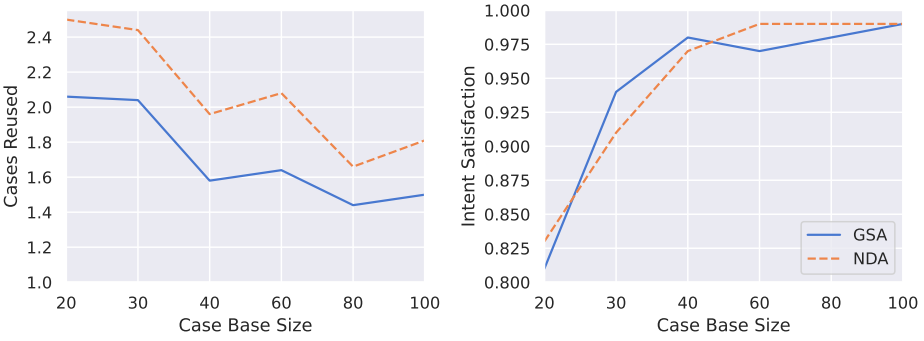

Fig. 9. Average number of nearest cases used for the reuse operation with the corresponding intent satisfaction for different case base sizes.

both GSA and NDA have identical intent satisfaction measures except that NDA continues to reuse more case than GSA as shown in Figure 9. In general, the GSA approach uses fewer cases to achieve solutions with relatively high intent

satisfaction compared to NDA for all case base sizes. The ability to reuse a smaller number of cases to construct a good solution is crucial in situations where the sparsity of the case base is a significant concern.

5 Conclusion

In conclusion, this paper proposed a novel approach to enhance problem-solving efficiency through the reuse of case solutions. Our approach called failure-driven transformational case reuse of explanation strategies involves transforming sub-optimal solutions by utilising relevant components from nearest neighbours in sparse case bases. Our approach uses failures as a starting point for generating new solutions, resulting in solutions that address the failure. We compared different approaches for reusing solutions of the nearest neighbours and empirically evaluated whether the transformed solutions meet the required explanation intents. The proposed approach has the potential to improve the effective reuse of relevant cases and facilitate the identification of appropriate adaptation strategies for new problems, especially in case bases that exhibit sparsity and complexity in case solutions.

The proposed reuse technique is not limited to its current domain of application and can be extended to other domains that use similar complex structures, such as game development and robotics, where behaviour trees are widely used. Our future work will focus on developing a generalised version of the reuse technique in CloodCBR to enable its broader application. This will include incorporating options for different use cases, such as allowing duplicates in the second element of the matching pairs (i.e., different explanation needs are satisfied by the same explainer), to enhance its flexibility and adaptability to a wider range of scenarios. Also, the intent satisfaction index used in the experiments measures one of the desirable features of the explanations, but it may not be sufficient to measure the quality of the solutions. We will explore methods to measure the quality of solutions such as conducting a study with real cases and users.

References

1. Colledanchise, M., Ögren, P.: Behavior trees in robotics and AI: An introduction. CRC Press (2018)
2. Dubins, L.E., Freedman, D.A.: Machiavelli and the gale-shapley algorithm. The American Mathematical Monthly **88**(7), 485–494 (1981)
3. Hanney, K., Keane, M.T.: Learning adaptation rules from a case-base. In: Advances in Case-Based Reasoning: Third European Workshop EWCBR-96 Lausanne, Switzerland, November 14–16, 1996 Proceedings 3. pp. 179–192. Springer (1996)
4. Jalali, V., Leake, D., Forouzandehmehr, N.: Learning and applying case adaptation rules for classification: An ensemble approach. In: IJCAI. pp. 4874–4878 (2017)
5. Liao, C.K., Liu, A., Chao, Y.S.: A machine learning approach to case adaptation. In: 2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). pp. 106–109. IEEE (2018)

6. Lowe, D.G.: Similarity metric learning for a variable-kernel classifier. Neural computation **7**(1), 72–85 (1995)
7. McSherry, D.: An adaptation heuristic for case-based estimation. In: Advances in Case-Based Reasoning: 4th European Workshop, EWCBR-98 Dublin, Ireland, September 23–25, 1998 Proceedings 4. pp. 184–195. Springer (1998)
8. Nkisi-Orji, I., Palihawadana, C., Wiratunga, N., Corsar, D., Wijekoon, A.: Adapting semantic similarity methods for case-based reasoning in the cloud. In: Case-Based Reasoning Research and Development: 30th International Conference, ICCBR 2022, Nancy, France, September 12–15, 2022, Proceedings. pp. 125–139. Springer (2022)
9. Nkisi-Orji, I., Wiratunga, N., Palihawadana, C., Recio-García, J.A., Corsar, D.: Clood cbr: Towards microservices oriented case-based reasoning. In: International Conference on Case-Based Reasoning. pp. 129–143. Springer (2020)
10. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019)
11. Schoonderwoerd, T.A., Jorritsma, W., Neerincx, M.A., Van Den Bosch, K.: Human-centered xai: Developing design patterns for explanations of clinical decision support systems. International Journal of Human-Computer Studies **154**, 102684 (2021)
12. Sokol, K., Flach, P.: One explanation does not fit all: The promise of interactive explanations for machine learning transparency. KI-Künstliche Intelligenz **34**(2), 235–250 (2020)
13. Wettschereck, D., Aha, D.W.: Weighting features. In: Case-Based Reasoning Research and Development: First International Conference, ICCBR-95 Sesimbra, Portugal, October 23–26, 1995 Proceedings. pp. 347–358. Springer (2005)
14. Wijekoon, A., Corsar, D., Wiratunga, N.: Behaviour trees for conversational explanation experiences. arXiv preprint arXiv:2211.06402 (2022)
15. Wilke, W., Vollrath, I., Althoff, K.D., Bergmann, R.: A framework for learning adaptation knowledge based on knowledge light approaches. In: Proceedings of the fifth german workshop on case-based reasoning. pp. 235–242 (1997)
16. Ye, X., Leake, D., Jalali, V., Crandall, D.J.: Learning adaptations for case-based classification: A neural network approach. In: Case-Based Reasoning Research and Development: 29th International Conference, ICCBR 2021, Salamanca, Spain, September 13–16, 2021, Proceedings 29. pp. 279–293. Springer (2021)