'What Is the Next Step?' Supporting Architectural Room Configuration Process with Case-Based Reasoning and Recurrent Neural Networks

Viktor Eisenstadt,[1] **Klaus-Dieter Althoff**[1 2]
[1]University of Hildesheim, Institute of Computer Science, Samelsonplatz 1, 31141 Hildesheim, Germany
[2]German Research Center for Artificial Intelligence (DFKI), Trippstadter Strasse 122, 67663 Kaiserslautern, Germany
viktor.eisenstadt@uni-hildesheim.de, klaus-dieter.althoff@dfki.de

Abstract

This paper presents the first results of the research into AI-based support of the room configuration process during the early design phases in architecture. Room configuration (also: room layout or space layout) is an essential stage of the initial design phase: its results are crucial for user-friendliness and success of the planned utilization of the architectural object. Our approach takes into account different possible actions of the configuration process, such as adding, removing, or (re)assigning of the room type. Its mode of operation is based on specific process chain clusters, where each cluster represents a contextual subset of previous configuration steps and provides a recurrent neural network trained on this cluster data only to suggest the next step, and a case base that is used to determine if the current process chain belongs to this cluster. The most similar cluster then tries to suggest the next step of the process. The approach is implemented in a distributed CBR framework for support of early conceptual design in architecture and was evaluated with a high number of process chain queries to prove its general suitability.

Introduction

The early stage of conceptual design in architecture consists of a number of sub-phases, where the task of constructuon of a proper room configuration is one of the most crucial sub-phases, as it is aimed at setting up the further room layout development for the future architectural object. Nowadays, the room configuration process is mostly based on the own experience of the architect, technical literature, or advice of colleagues. That is, the concrete records of particular steps of room configuration processes of other architects are neither considered by architects nor permanently saved and interpreted for further use in any professional software for architectural design. Therefore, an intelligent, computer-aided system that makes use of such step records and can suggest a proper next step can provide an additional source of help and inspiration during the room layout creation process. The biggest advantages of such system would be the minimization of time needed to determine the next configuration step and contextual suggestions, i.e., suggestions provided only for the given task domain (e.g., office building floor plans).

In this paper, we present an approach for generation of such suggestions by means of applying a combination of methods of case-based reasoning (CBR), clustering, and recurrent neural networks (RNN). The approach is an essential part of MetisCBR (Ayzenshtadt et al. 2016a), a distributed CBR framework for support of early phases in architectural design, and was conceptualized and implemented to work as a native module of this framework.

Related Work

The research connection between artificial intelligence (AI) and computer-aided architectural design (CAAD) has a long history and resulted in a multitude of approaches for AI-based support of architectural design: for example, case-based design (CBD), mainly for retrieval or explanation, and ANN-based systems, mainly for analysis or generation of building designs. Being very knowledge-intensive entities, architectural designs provide a broad base for construction of cases of different type, such as image-based, graph-based (Langenhan et al. 2013), or attribute-value based (Ayzenshtadt et al. 2015). The latter one is used in MetisCBR and is inferred from graph-based representations of floor plans. A short description of the framework is provided in the next section. A survey of pre-existing/earlier developed framework-related methodologies and tools (Richter, Heylighen, and Donath 2007) provides a selection of the most influencial CBD-CAAD approaches. An overview of the current problem fields of CBR in architecture was published as well (Richter 2013). One of the most relevant recently developed approaches uses deep learning to extract semantic features from floor plan images (Sharma et al. 2017). A combination of CBR, multi-agent systems (MAS), and ANNs was used to optimize energy management in buildings (using an example of an office building) (González-Briones et al. 2018). A specific case of floor plan generation with RNNs uses block-based feature vectors for room layout and connection generation (Bayer, Bukhari, and Dengel 2017).

MetisCBR

MetisCBR (see Figure 1) is a distributed AI system that applies *case-based retrieval* and *pattern-based explanation* methods to support the early phases of architectural conceptual design. The CBR retrieval component of the framework looks for architectural designs (floor plans) suitable for improvement of the currently developed design in a case base of previous designs. This functionality employs a number of *retrieval containers*, where each container is responsible for resolving of the floor plan query with one of the specific search patterns, the *semantic fingerprints of architecture* (Langenhan and Petzold 2010). The semantic fingerprints represent architectural concepts, such as *room and relation counts*, *adjacency and accessibility of rooms*, or *availability of natural light*, and can be selected by the user. That is, each container is responsible for a sub-query only. The single similarity values achieved for the sub-queries are then amalgamated to produce the overall similarity value.

The retrieval results are then enhanced with information added by the explanation component of the framework, the *Explainer* (Eisenstadt et al. 2018). This component is based on the paradigm of *explanation patterns* (Cassens and Kofod-Petersen 2007) and tracks the system behavior during the retrieval process. Its main task is to find answers to questions of *why* a specific result was included in the result set (justification pattern) and *how* exactly the system found this solution (transparency pattern), and to find out if it is relevant to ask the user for further data (relevance pattern). A further task of this component is to add contextual information that shows how the results are related to each other.

The framework was extended to a flexibility- and process-oriented CBR system (Ayzenshtadt et al. 2017). Together with other floor plan retrieval techniques, MetisCBR was examined in a number of comparative evaluations (Sabri et al. 2017) and (Ayzenshtadt et al. 2016b).

Approach for Suggestion of the Next Room Configuration Step

In this section, we present our methodology for intelligent support of the room configuration process during the early conceptual phases in architectural design. We describe the general structure of the *Suggester* (MetisCBR's system module that implements the room configuration support approach), present each phase of generation of suggestions in detail, and explain the core concept of the configuration process, the *process chain*.

General Overview

Figure 1 shows the general structure and mode of operation of MetisCBR's component for suggestion of the next configuration step during the interaction with the system. Identically to the other system modules, the *Suggester* is a sub-MAS of the entire multi-agent-based architecture of the framework. Currently, four agents govern the corresponding suggestion generation process:

- *Suggestion Preparation Agent* – This agent prepares, i.e., separates, the process chain query for later resolving of its main parts: the actions, positions, and relations. It also performs the CBR retrieval to determine which process chain cluster provides the most suitable configuration data for the current query and the corresponding suggestion.
- *Next Action Suggestion Agent* – The task of this agent is to query the RNN of the cluster selected by the preparation agent and to return as many possible actions for the process chain continuation as requested (the exact amount of action suggestions can be pre-configured).
- *Position Suggestion Agent* – If the result of action suggestion is to add a new room, this agent tries to determine the concrete position of this room in the configuration.
- *Relation Suggestion Agent* – As continuation of position suggestion, the task of this agent is to find out which connection types (i.e., connection by doors or passages) can be used to connect the suggested room to the rooms provided by the position suggestion agent.

The Process Chain

The core concept of our approach is the so-called *process chain*. Each process chain is a directed acyclic graph (DAG), where each node represents an *action* of the room configuration process. The action can be one of the following:

- **a** – Add a new room
- **r** – Remove an existing room
- **f** – Reshape, change form (polygon) of an existing room
- **t** – Change the type (functionality) of the room

Actions are accompanied by the *abstraction level* of the room, which can be either **0** (abstract, i.e., a room without particular shape) or **1** (non-abstract, i.e., a room with shape specified), and a *room type*, which can be one of the following: **L** (living room), **S** (sleeping room), **K** (kitchen), **T** (toilet), **W** (working), **B** (bath), **C** (corridor), **P** (storage), or **R** (room with no specific type, i.e., a placeholder room).

Each action is represented by a full action signature, e.g., **a1K** for *add a kitchen and specify its shape*.

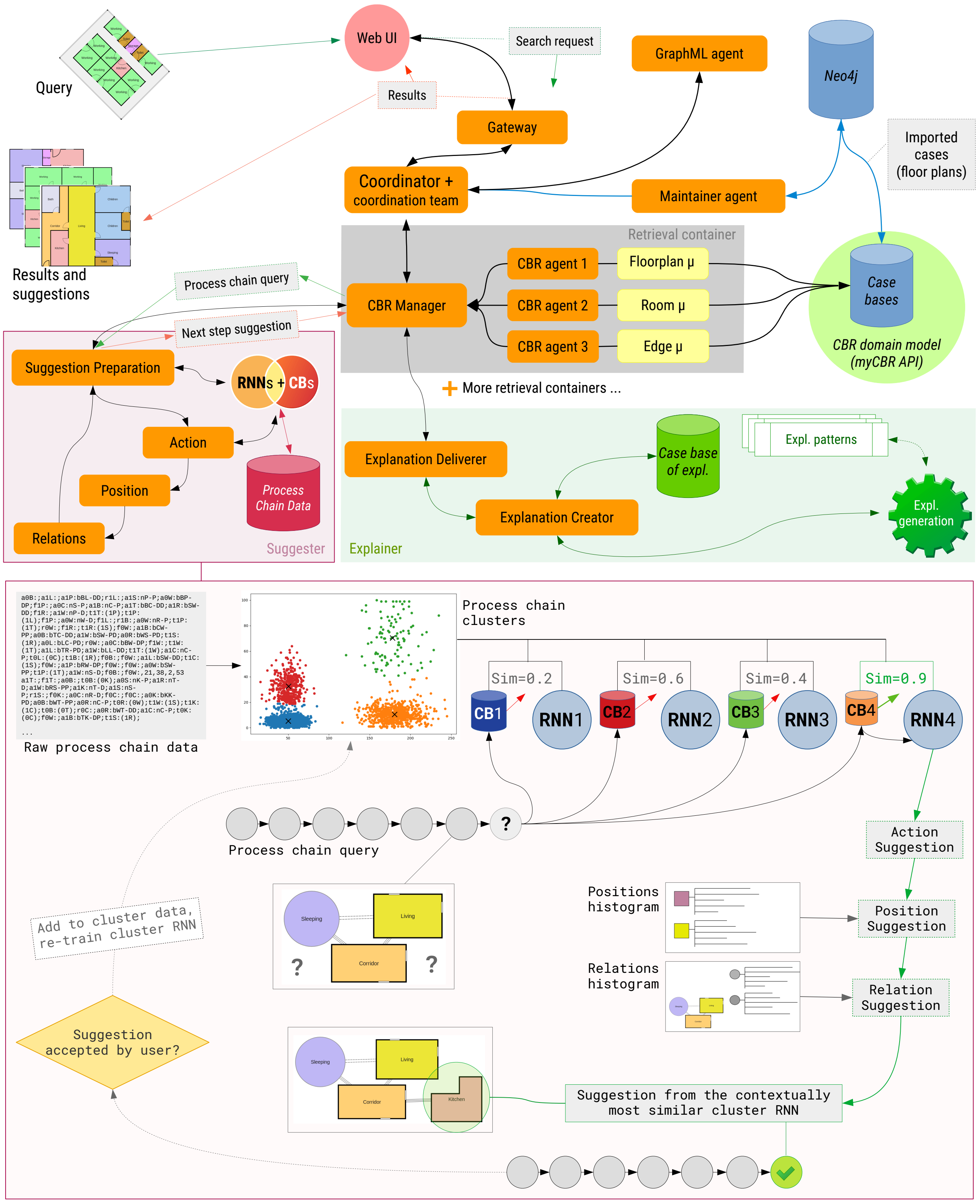
If action is an **a**, then it is enriched with additional information about positioning within the configuration and relations/connections to the neighbouring rooms. Positioning is represented by a *position type* which can be either **b**, if the room should be placed between multiple rooms, or **n**, if the room should be placed next to one specific room. Relations represent the connection type(s) of the room from the action to the rooms from the positioning, and can be either **D** for a *door* or **P** for a *passage*.

For example, the signature **a0L:bWS-DD** stands for *add an abstract living room between Working and Sleeping, connect it with doors to both of them.*

Additionally, when the user asks the system for a step suggestion, each process chain is enriched with meta data that represents the current *room count* and *edge count* of the floor plan, the *number of semantic fingerprints* applied for the current search, and the *current number of actions* in the chain. Figure 2 demonstrates a sample process chain DAG.

Process Chain Clusters

To proceed with the actual suggestion process, the properly formatted data for this process needs to be created from the previously recorded chains. The first step of the creation of the properly formatted data for suggestion generation is the initial separation of the complete process chain dataset into specific subsets, so-called *process chain clusters*. To assign



```
a0B:;a1L:;a1P:bBL-DD;r1L:;a1S:nP-P;a0W:bBP-
DP;f1P:;a0C:nS-P;a1B:nC-P;a1T:bBC-DD;a1R:bSW-
DD;f1R:;a1W:nP-D;t1T:(1P);t1P:
(1L);f1P:;a0W:nW-D;f1L:;r1B:;a0W:nR-P;t1P:
(1T);r0W:;f1R:;t1R:(1S);f0W:;a1B:bCW-
PP;a0B:bTC-DD;a1W:bSW-PD;a0R:bWS-PD;t1S:
(1R);a0L:bLC-PD;r0W:;a0C:bBW-DP;f1W:;t1W:
(1T);a1L:bTR-PD;a1W:bLL-DD;t1T:(1W);a1C:nC-
P;t0L:(0C);t1B:(1R);f0B:;f0W:;a1L:bSW-DD;t1C:
(1S);f0W:;a1P:bRW-DP;f0W:;f0W:;a0W:bSW-
PP;t1P:(1T);a1W:nS-D;f0B:;f0W:,21,38,2,53
a1T:;f1T:;a0B:;t0B:(0K);a0S:nK-P;a1R:nT-
D;a1W:bRS-PP;a1K:nT-D;a1S:nS-
P;r1S:;f0K:;a0C:nR-D;f0C:;f0C:;a0K:bKK-
PD;a0B:bWT-PP;a0R:nC-P;t0R:(0W);t1W:(1S);t1K:
(1C);t0B:(0T);r0C:;a0R:bWT-DD;a1C:nC-P;t0K:
(0C);f0W:;a1B:bTK-DP;t1S:(1R);

...
```

Figure 1: General structure of MetisCBR, with the configuration support module, the Suggester, included.

the chain data to the proper cluster, we use the K-Means clustering method that takes the process chain meta data (see Figure 2) as feature vector input. The number of clusters is arbitrary and can be specified during the clustering process.

The *reason for* creation of specific data clusters and the *decision against* the usage of the complete process chain dataset, is the *specific structure of architectural data*: in our previous research (Ayzenshtadt et al. 2017), we found out that different conexts of architectural design tasks and floor plan classes can help identify the most helpful, i.e., contextually suitable, design to continue the development of the floor plan at hand. These different contextual classes can represent, for example, floor plans with the number of connections that is slightly lower or marginally higher than the number of rooms (Sparse Connections Context) or a certain room type has a high number of occurences, i.e., dominates the floor plan (Room Type Dominance Context). Based on results of this research, we transferred this contextual design classification approach to the contextual data separation of the configuration support module. I.e., our assumption is, that for our main goal – *find the most exact and suitable suggestion for the next step of the room configuration process* – contextual clusters can be a proper approach too. By dividing the complete dataset into smaller subsets, we intend to query the cluster with the highest chances to provide the best contextual suggestion.

After the separation of the dataset into contextual clusters, we train each cluster data with a separate recurrent neural network (RNN) and set up a corresponding case base of *context footprints* that will be used for determination of the most suitable cluster for suggestion generation.

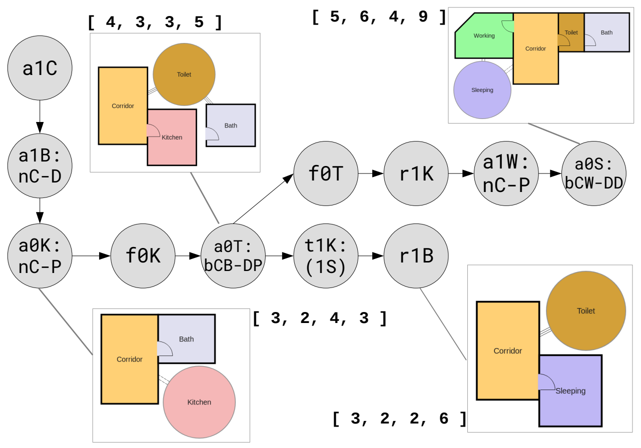


Figure 2: An example of a process chain DAG. The rectangles represent the floor plan's configuration state at the given step. Nodes represent the actions and contain the full action signature: e.g., **a0K:nC-P** for *add an abstract kitchen next to the corridor, connect them with a door* or **f0T** for *add shape to the toilet*. Numbers in the square brackets represent the meta data of the configuration at the given step (room count, edge count, number of semantic fingerprints used in the search, and the current step count accordingly). Floorplans are constructed with Metis WebUI (Bayer et al. 2015).

Training of Cluster Data with RNN

Over the last decades, artificial neural networks became an established tool for a multitude of prediction-related tasks, including tasks where the input data is a sequence of arbitrary length that can represent events, e.g., click events on a webpage to predict the next user action, or words, e.g., words in a sentence that needs to be completed. The most established ANN type for this task (also referred as *sequence learning* (Sutskever, Vinyals, and Le 2014)) are the recurrent neural networks, that, e.g., can be used in the autocompletion feature of keyboards of mobile operating systems (Valkov 2017).

RNNs differ from other ANN types in the way in which they handle the dependencies between the network states at the current and previous time steps. RNNs are based on a premise that states of the previous time steps can be remembered by the corresponding cells of the network layer, thus providing the network with the possibility to record dependencies over time and use this feature to predict the most likely next event, i.e., the continuation of the sequence. Nowadays, different methods exist for the 'remembering' feature of an RNN cell. The most widely applied ones are *LSTM* (Long Short-Term Memory) (Hochreiter and Schmidhuber 1997), and *GRU* (Gated Recurrent Unit) (Cho et al. 2014) that gained popularity within the last years.

Based on the facts described above, selecting RNNs for the task of prediction of the next action during the room configuration process, is a logical step as the process chains are sequences of arbitrary length too. They are ordered in a sentence-like way as well, and basically represent the user events on a user interface for room configuration. Thus, our assumption is that the RNN-based step prediction approach is the most suitable one for the research task of this work.

Case Bases of Context Footprints

As mentioned above, our complete process chain dataset for room configuration support is separated into different contextual subsets before the RNNs start to train on them. To identify which of these clusters is the most suitable one for prediction of the next step, we apply a case-based retrieval approach that consists of the following steps:

1. Select randomly a subset of process chains from the cluster and extract their meta data.
2. Turn each of the meta data entries into a case and save it into the cluster's case base, i.e., create the first version of the so-called *context footprint*.
3. *When the next process chain query arrives:*
   (a) Extract its meta data and turn it into the query for all case bases for all clusters.
   (b) Find out which cluster case base returns the highest similarity value:

$$v = \frac{1}{4}\sum_{f=1}^{4} Sim_f, \quad f \in F$$

   Where F is the set of features f from the meta data (room count, edge count, semantic fingerprint count, steps count) for both query and case.

(c) Forward the actual process chain query to the RNN of cluster with the highest similarity.

(d) *If later, after the complete suggestion has been produced, the user accepts the suggested step:*

   i. Save the sequence in the cluster data, i.e., append the suggested step to the existing sequence.

   ii. Re-train the corresponding RNN.

   iii. Remove all cases from the *winning* case base, except the one that has returned the actual higest similarity (this case will be persisted, but it can be removed too if other cases from this cluster will be selected more often, i.e., the *winning* cases will be ranked).

   iv. Goto 1 (for this cluster only).

The approach we use for determination of the most suitable/similar cluster is a combination of the two well-known CBR approaches: *footprint similarity* (Smyth and McKenna 1999) and *instance-based learning* (Aha, Kibler, and Albert 1991) with the IB3 algorithm. From IB3, we use the principle of a case base of (auto-selected) 'golden standard' cases. From the footprint similarity approach we transferred the principle of the optimal footprint coverage: by persisting and ranking the *winning* cases and removing the ones with lesser similarity we will eventually create a *competence set* that represents the cluster context in the best way possible.

Querying the Context RNN

As shown in Figure 1 and explained in the previous sections, only the RNN, whose case base contains the most similar case to the query meta data, is queried. As a rule of thumb, only a certain number of the last steps in the sequence is used to query the RNN and to get action suggestion(s). The reason for this is the potential to get a wider variety of possible configuration continuations, thus, optionally, present the user with a number of alternative steps too.

Position Suggestion

As mentioned above, if the outcome of the action suggestion is to add a room to the existing configuration, position of the suggested room within the configuration and relation type(s) for connection (see next section) can be determined too. To accomplish this task of finding the best position, we first create a histogram for position entries of each action found in the initial process chain dataset. To add position for the current action suggestion, we first analyze this action's histogram and remove entries where at least one room is not available in the current room configuration. From the rest of the histogram entries, an entry with the highest number of usages is selected, added to the full suggestion signature, and forwarded to the relation suggestion component/agent. If multiple suggestions were requested, we take as many often used positions as needed, in descending order.

Relation Suggestion

The relation suggestion component provides a functionality that is similar to the position suggestion component: it creates an instant histogram of all room connections for each room available in the floor plan configuration and provides the most used ones for as many rooms and relations as requested. This is the only functionality of the whole methodology that does not depend on the apriori available data to create a suggestion. Instead, it operates entirely on the data from the current room configuration status and depends only on the steps of the current user session.

Evaluation

To initally evaluate the room configuration support methodology described in the previous sections of this paper, we decided to perform an automated quantitative experiment, whose main goal was to prove the general suitability of the approach for the real-world use in MetisCBR, for example, for future user studies.

As no functionality to record the user's room configuration steps is available in any of the user interfaces of the framework, it has been decided to generate the data needed for the experiment. The data generation has been performed with consideration of a number of constraints that were necessary to imitate the real behavior of the user during the configuration process. For example, the positioning generation should consider which rooms are available in the current configuration, or, removing or reshaping of the rooms should update the current (i.e., temporarily saved) room configuration to avoid conflicts, such as erroneous positioning.

Setting

To conduct the quantitative experiment, we first generated an initial amount of 1000000 process chains, where each chain contained between 10 and 60 configuration steps. Each chain was assigned to one of the 100 contextual clusters using the K-Means clustering method. For each of these clusters, we extracted an initial amount of 50 chains to create the corresponding context footprint case bases. After that, each of the 100 subsets (i.e., each cluster chain data) was fed into a separate RNN with the following configuration:

- 3 deep layers, each with 512 GRU cells
- Maximum of 7 inputs (i.e, previous steps) and maximum of 7 softmax-produced outputs (i.e., next actions)

To produce suggestions, 1000 query chains were generated, using the generation method used for the chain dataset.

Results

The main goal of the analysis of the results was to find out the percentage of valid chain continuations. In the context of this experiment, a *valid continuation* is a suggestion that can *potentially* be accepted by a user in a real-world scenario. For example, if it has been suggested by the system to add a new room *and* it was possible to determine a position and corresponding relations, then the suggestion was marked as 'valid', or if the system suggested to remove, reshape, or change the type of the room, but no such room/room type was available in the current configuration, then the suggestion was marked as 'non-valid'. Per query, a maximum of 2 suggestions has been set (second suggestion run only for queries with no outcome on the first run) with a maximum of 4 retries to find a proper position. On the first run already, we could count 86% of valid/accepted suggestions, and could increase it to $\approx 97\%$ on the second run. From the CBR perspective, we were interested in similarity values of cluster case base selection, i.e., all values that the Suggester used to decide which case base is the most suitable one. Overall, for all selection processes, the minimum value of 0.74 and the maximum of 0.99 could be achieved.

Conclusion and Future Work

In this paper, we presented an approach for support of the room configuration process in architecture. The approach uses CBR, RNNs, and clustering to identify the most suitable continuation of room layout in the current context. It is planned to conduct a number of user studies to examine the potential of the methodology for the target group: the architects and architecture students.

References

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine learning* 6(1):37–66.

Ayzenshtadt, V.; Langenhan, C.; Bukhari, S. S.; Althoff, K.-D.; Petzold, F.; and Dengel, A. 2015. Distributed domain model for the case-based retrieval of architectural building designs. In Petridis, M.; Roth-Berghofer, T.; and Wiratunga, N., eds., *UKCBR-2015, December 15-17, Cambridge, UK*.

Ayzenshtadt, V.; Langenhan, C.; Bukhari, S. S.; Althoff, K.-D.; Petzold, F.; and Dengel, A. 2016a. Thinking with containers: A multi-agent retrieval approach for the case-based semantic search of architectural designs. In Filipe, J., and van den Herik, J., eds., *8th International Conference on Agents and Artificial Intelligence (ICAART-2016), February 24-26, Rome, Italy*. SCITEPRESS.

Ayzenshtadt, V.; Langenhan, C.; Roith, J.; Bukhari, S. S.; Althoff, K.-D.; Petzold, F.; and Dengel, A. 2016b. Comparative evaluation of rule-based and case-based retrieval coordination for search of architectural building designs. In Goel, A.; Roth-Berghofer, T.; and Diaz-Agudo, B., eds., *Case-based Reasoning in Research and Development. ICCBR-2016, October 31 - November 2, Atlanta, Georgia, USA*. Springer, Berlin, Heidelberg.

Ayzenshtadt, V.; Langenhan, C.; Bukhari, S.; Althoff, K.-D.; Petzold, F.; and Dengel, A. 2017. Extending the flexibility of case-based design support tools: A use case in the architectural domain. In *Case-Based Reasoning Research and Development, ICCBR-2017, Trondheim, Norway, June 26-28, 2017, Proceedings*, volume 10339, 46. Springer.

Bayer, J.; Bukhari, S.; Dengel, A.; Langenhan, C.; Althoff, K.-D.; Petzold, F.; and Eichenberger-Liwicki, M. 2015. Migrating the classical pen-and-paper based conceptual sketching of architecture plans towards computer tools - prototype design and evaluation. *11th IAPR International Workshop on Graphics Recognition - GREC'15, Nancy, France*.

Bayer, J.; Bukhari, S. S.; and Dengel, A. 2017. Floor plan generation and auto completion based on recurrent neural networks. In *ICDAR 2017, International Workshop on Graphics Recognition (GREC)*, pp. 49–50.

Cassens, J., and Kofod-Petersen, A. 2007. Designing explanation aware systems: The quest for explanation patterns. In *ExaCt*, 20–27.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Eisenstadt, V.; Espinoza-Stapelfeld, C.; Mikyas, A.; and Althoff, K.-D. 2018. Explainable distributed case-based support systems: Patterns for enhancement and validation of design recommendations. In Cox, M. T.; Funk, P.; and Begum, S., eds., *Case-Based Reasoning Research and Development*, 78–94. Cham: Springer International Publishing.

González-Briones, A.; Prieto, J.; De La Prieta, F.; Herrera-Viedma, E.; and Corchado, J. M. 2018. Energy optimization using a case-based reasoning strategy. *Sensors* 18(3):865.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Langenhan, C., and Petzold, F. 2010. The fingerprint of architecture-sketch-based design methods for researching building layouts through the semantic fingerprinting of floor plans. *International electronic scientific-educational journal: Architecture and Modern Information Technologies* 4:13.

Langenhan, C.; Weber, M.; Liwicki, M.; Petzold, F.; and Dengel, A. 2013. Graph-based retrieval of building information models for supporting the early design stages. *Adv. Engineering Informatics* 27(4):413–426.

Richter, K.; Heylighen, A.; and Donath, D. 2007. Looking back to the future – an updated case base of case-based design tools for architecture. *Knowledge Modelling-eCAADe*.

Richter, K. 2013. What a shame-why good ideas can't make it in architecture: A contemporary approach towards the case-based reasoning paradigm in architecture. In *FLAIRS Conference*.

Sabri, Q. U.; Bayer, J.; Ayzenshtadt, V.; Bukhari, S. S.; Althoff, K.-D.; and Dengel, A. 2017. Semantic pattern-based retrieval of architectural floor plans with case-based and graph-based searching techniques and their evaluation and visualization. In *6th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2017), February 24-26, Porto, Portugal*.

Sharma, D.; Gupta, N.; Chattopadhyay, C.; and Mehta, S. 2017. Daniel: A deep architecture for automatic analysis and retrieval of building floor plans. In *ICDAR 2017 Proceedings*, volume 1, 420–425.

Smyth, B., and McKenna, E. 1999. Footprint-based retrieval. In *Case-Based Reasoning Research and Development*. Springer. 343–357.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Valkov, V. 2017. Making a predictive keyboard using recurrent neural networks - tensorflow for hackers, part v. http://curiousily.com/data-science/2017/05/23/tensorflow-for-hackers-part-5.html, visited on 19.11.2018.