Master's Thesis

Development of a Modular Software Framework for Supporting Architects During Early Design Phases

by
Johannes Bayer

July 31, 2017

supervised by
Prof. Dr. Andreas Dengel
and
Dr. Ing. Syed Saqib Bukari

Department of Computer Science
University of Kaiserslautern

Contents

Ich versichere hiermit, dass ich die vorliegende Masterarbeit mit dem Thema "Development of a Modular Software Framework for Supporting Architects During Early Design Phases" selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

Ort, Datum

Johannes Bayer

Abstract

Early design phases in architecture deal with the conceptualization of a building. During these phases, a rough floor plan layout is designed by an architect based on a high-level description given by a contractor or customer. Traditionally, these phases involve a lot of monotone and repetitive labor. For example, manual research in architectural libraries or dedicated magazines is carried out in order to retrieve reference concepts that may serve as inspiration or to probe the feasibility of the project. Likewise, some aspects of a floor plan design are highly creative and some parts remain rather predictable. One established working method in architecture for turning a high-level description into a specific floor plan layout is the so-called *room schedule*, in which a set of individual rooms is the focus of interest.

Based on this room schedule working method, a smart framework is developed to assist architects with their work: A sketch editor allows for specifying a floor plan concept with a varying degree of abstraction, thus allowing the user to specify every aspect of a floor plan concept as abstract of specific as desired. The sketch editor follows the room schedule working method and aims to support the architect during the entire early design phase.

Simultaneously, this sketch editor serves as an input tool for search queries with which the user can search for similar floor plan concepts. The presented framework allows for integrating an arbitrary number of such retrieval systems that utilize a dedicated floor plan database. Currently, three different retrieval systems are integrated, one of them is presented in greater detail here. These retrieval systems rely on so-called semantic fingerprints, which are graph-based abstractions of floor plan concepts. Hence, subgraph matching is employed to these graph-based abstractions of floor plans. Likewise, these semantic fingerprints allow the user to control the retrieval process. In order to accomplish explainability in the result finding, a mapping function is incorporated in the framework that calculates for all found results, how individual rooms in the search query map to individual rooms in the search results.

Finally, the sketch editor is connected to a neural network-based predictor that generates automatic suggestions for solving creative problems. These suggestions may perform entire design steps. This helps the user to avoid repetitive and predictable actions like completing a floor plan layout. Likewise, it offers inspiration and provides templates that can be refined by the user. In such a setting, the user and the artificial neural network are in a loop, both manipulating the sketch.

In order to evaluate the proposed solution, individual aspects of the framework are tested by different means to demonstrate their usefulness. The user interface is tested by user study that compared it to the traditional approach. The retrieval system is tested by both a stress test and qualitatively analysis. Finally, the design suggestion is both evaluated by established means of artificial intelligence and examined qualitatively.

Zusammenfassung

Frühe Phasen architektonischen Entwerfens befassen sich mit der Konzeptualisierung eines Gebäudes. Dabei wird ein grober Grundriss-Entwurf ausgehend von einer sehr abstrakten Beschreibung entwickelt, welche üblicherweise vom Bauunternehmer oder dem Endkunden stammt. Diese Phasen enthalten traditionell sehr arbeitsintensive und monotone Aufgaben. So wird beispielsweise oft eine Literaturrecherche in Architektur-Bibliotheken und spezialisierten Magazinen durchgeführt, um nach Referenzen zu suchen, die als Inspiration dienen können und die Machbarkeit des Projektes zeigen sollen. Ebenso sind zwar manche Aspekte im Entwurfsprozess hochgradig kreativ, andere hingegeben eher vorhersagbar und trivial. Eine etablierte Arbeitsmethode, um eine abstrakte Projektbeschreibung in einen Grundriss-Entwurf zu überführen ist das sogenannte *Raumprogramm*, bei dem eine Gruppe von einzelnen Räumen im Vordergrund steht.

Basierend auf dieser Raumprogramm-Arbeitsmethode wird ein intelligentes Framework entwickelt, um Architekten bei ihrer Arbeit zu assistieren: Ein Entwurfsprogramm erlaubt es, Grundrisse mit variablem Abstraktionsgrad einzugeben, wobei jeder Aspekt des Entwurfs so abstrakt oder spezifisch wie gewünscht angegeben werden kann. Dieses Entwurfsprogramm folgt der Idee des Raumprogramms und zielt darauf ab, den Architekten über die gesamte frühe Phase des Entwurfs zu begleiten.

Gleichzeitig dient dieses Entwurfsprogramm als Schnittstelle zu einem Suchsystem, dass automatisch nach ähnlichen Entwürfen sucht. Das präsentierte Framework erlaubt dabei die Integration einer beliebigen Anzahl von internen Suchverfahren. Zur Zeit sind drei solche Module ins System integriert, von denen auf eines im Detail eingegangen wird. Diese Suchverfahren basieren auf sogenannten semantischen Fingerabdrücken, welche graphbasierte Abstraktionen von Grundriss-Konzepten darstellen. Folglich kommt Subgraph-Matching beim Suchprozess zum Einsatz. Die semantischen Fingerabdrücke können vom Nutzer gewichtet werden, um die Suche zu beeinflussen. Eine in das Framework integrierte Abbildungsfunktion einzelner Räume der Sucheingabe auf Suchergebnisse hilft dabei dem Nutzer, den Findungsprozess des Suchsystems nachzuvollziehen.

Schlussendlich ist das Entwurfsprogramm mit einer auf neuronalen Netzwerken basierenden Entwurfs-KI verbunden, um die automatische Lösung kreativer Probleme anzustreben. Dieser Mechanismus kann ganze Entwurfsschritte ausführen. Dies hilft dem Nutzer, monotone Entwurfschritte zu vermeiden. Dies kann ebenfalls der Inspiration dienen und die vom Mechanismus erstellen Entwürfe können vom Nutzer verfeinert werden. In dieser Situation sind Architekt und KI miteinander über den Entwurf verbunden.

Das präsentierte Framework wird anhand einer Reihe verschiedener Experimente auf seine Tauglichkeit untersucht. Die Nutzerschnittstelle wird durch eine Nutzerstudie untersucht, das Suchsystem wird durch einen Stresstest und qualitativ untersucht. Die Entwurfs-KI wird durch etablierte Methoden Der KI und qualitativ untersucht.

1. Introduction

Paper is dead without words,
Ink idle without a poem,
All the world dead without stories

– Tuomas Holopainen

1.1. Early Design Phases in Architecture

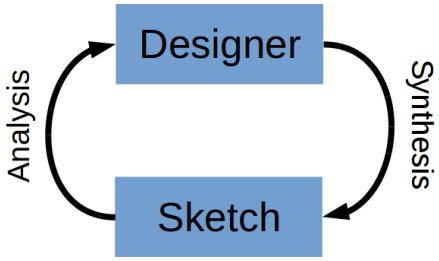


Figure 1.1.: Iterative Design Process using Sketches (adapted from [20]).

While late design phases in architecture are rather technical (like conducting a structural analysis of a design or getting approval by building department), early design phases involve a major part of the creative work. Several attempts have been made to model the process of architectural design (e.g. [12], see [46] for an overview). As a general principle, the process of design is usually carried out in an iterative manner (see fig. 1.1). In this loop, the architect is the main actor and the sketch serves as an external memory extension only. The architect creates a sketch based on the actual design problem, experiences and multiple other constraints (e.g. economic restrictions, rules of the art). After creating an initial sketch, the sketch is analyzed to spot problems regarding the mentioned constraints (or at lest one of them is focused) and a refined version is generated. The process continues until a useful result is created.

1.1.1. The Room Schedule Working Method

The room schedule is a top down working method that is influenced by the way a building project is ordered by a contractor. A room schedule is a high-level specification of a building that usually comes from the customer or contractor and describes only a few aspects (e.g. "residential building for one family with two bathrooms and three

bedrooms"). Formally, it can consist of a simple list of room functions, but there might already also be other (not necessarily complete) restrictions like sizes of individual rooms and connections between rooms (e.g. "the kitchen should be placed next to the dining room and this connection should be traversable"). Obviously, the latter is better to be considered as a graph structure. Consequently, a room schedule is a rather vague concept, that can not be described by one data structure (lists, graphs and sets might however be appropriate for specific cases). Likewise, the degree of detail might range widely. Given a room schedule, the architect's task is to implement this specification as a floor plan while obeying given restrictions (like the established building practices). Different representations can be used in this stage, like so-called `bubble diagrams` in which each room is represented by a circular tag and the connections between them are indicated by lines. Likewise, free-style drawings can be used. This process is usually iterative, i.e. starting from a certain stage that is drawn on a first sheet of paper, it slightly refined on a second sheet, iterating until a certain stage is reached. In order to simplify this process, semitransparent paper is used in stacks, so that after a new sheet has been added, the existing sketch on the last sheet shines through the new sheet and allows for easy copying of aspects that should be equal in the old and new version.

1.1.2. Search for Architectural References

Initially, Architects may seek for external inspiration in order to complete their task. This process can be described as a research for references (or literature research). Such references can often be found in specialized literature. These pieces of literature may have solved a similar problem already, hence aspects may be reused for a given problem. Often, multiple sources may serve as inspiration.

1.1.3. Problems of the Traditional Workflow

Nowadays, the literature research is usually conducted manually and involves the consultation of different media (e.g. in specialized journals in paper form, sometimes only accessible in dedicated places like libraries). This process is very labor-intensive and involves monotone tasks like gathering the sources and evaluating them for significance for the ongoing building project. Likewise, the initial sketching itself is also labor intensive. Finally, paper-based drawings suffer from known problems: They have to be processed (e.g. stored/copied/edited/moved) manually and they can only be accessed by a limited amount of persons at the same time. When working with stacks of semitransparent paper, the stack's order can easily become messy and the process of redrawing by design requires repetitive, monotone actions.

1.2. Approach of this Thesis

The entire mentioned workflow could benefit from automation and digital media: the literature research for references that are needed is usually carried out in advance, the sketching itself as a process of recording concepts as well as the iterative process of

interpreting and modifying of these sketches. In order to accomplish this, an end-to-end system is developed that incorporated a set of functions to assist the human architect.

Sketches can be drawn in a dedicated user interface (UI). The UI is built with a focus on allowing to formulate a sketch (and every aspect of it) as abstract or specific as intended by the user following the room schedule working method. It starts with a number of rooms that can later be assigned sizes, functions, wall layouts and window and door positions. In general, the UI is room-oriented and uses established mouse/keyboard or pen-based interaction devices.

The search system can be used for automatically conducting a search for similar references based on the user's input. This search is conceptually based on abstractions of floor plans, the so-called *semantic fingerprints*. These abstractions mainly use mathematical structures of graphs to model floor plans where each room is a node in the graph. These hand-crafted semantic fingerprints embody architectural knowledge about the perception of floor plans in oder to make the similarity between floor plans computable. The user controls this retrieval process by determining the importance of individual fingerprints that are significant for him in the given situation. Different approaches were used to implement such a retrieval systems. In order to let the user understand the results and to reflect his/her work, the system explains the results by an illustrated mapping that relates the sketched rooms to the rooms in the search results.

In order to assist the user furthermore in his/her work, the system provides a function for proposing suggestions for different design steps. This function is implemented based on neural networks, that have been trained on a database of complete floor plans. It has to be invoked manually by the user, consequently the user still remains in control of the process.

1.3. Contribution

The contribution of this thesis to the scientific community can be summarized as follows:

- Development of a modular end-to-end system and framework for supporting architects during early design phases
- Development of a web-based user interface (WebUI) for accessing the framework
- Development of an index-based floor plan retrieval system
- Development of algorithm for explaining search results by mapping rooms in queries to rooms in results.
- Development of an design assisting system following the room schedule working method based on recurrent neural networks

This work makes use of preliminary studies that have been conducted in the course of the research project Metis, which has been funded by the Deutsche Forschungsgemeinschaft (DFG). Likewise, the main part of the research conducted for this thesis was also

conducted in the course of the research project Metis. Metis [8] has been conducted as a collaboration between the German Research Center for Artificial Intelligence (DFKI,[1]) and the Lehrstuhl für Architekturinformatik [7] at Technical University of Munich.

1.4. Structuring of this Thesis

The thesis at hand is structured as follows: After the current situation in the application domain and its related problem have been described in this chapter, chapter 2 describes the technical foundations used in the course of this thesis. Afterwards, similar work and approaches related to this thesis are explained in chapter 3. Then, the very methods to tackle the problem are outlined in chapter 4. After that, the proposed software solution is described in chapter 5. Different aspects of this solution are evaluated by different experiments in chapter 6. Finally, the gathered findings are summarized and the thesis is concluded by an overall résumé and incorporation of an outlook on the topic in chapter 7.

2. Background

This chapter introduces the technical and scientific foundations that are used in the thesis at hand. For each scientific branch that is utilized in this thesis, there is a section in this chapter. The first section introduces the mathematical concept of graphs that are employed in the floor plan retrieval function. Then, artificial neural networks are introduced which are needed in the creativity engine for generating design proposals. Finally, human computer interaction is briefly introduced as a scientific basis for the design and more importantly the analysis of the WebUI as a user interface.

2.1. Graphs

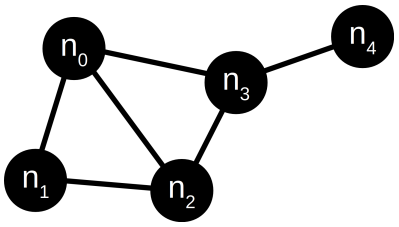


Figure 2.1.: Illustration of an Unlabeled Graph Consisting of 5 Nodes and 6 Edges.

Graphs[21] are a mathematical construct (see fig. 2.1 for an example illustration). The idea behind a graph is to model the relation between a set of objects. The objects in such a model are referred to as *nodes* and the relations are referred to as *edges*.

2.1.1. Definition of a Graph

In the following, a graph is formally defined as follows: Let $\mathcal{N} = \{n_0 \dots n_k\}$ be a final set of k nodes (nodes are also referred to as vertices in literature). Let furthermore $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ be a subset of the Cartesian product of $\mathcal{N}$ with itself. Then $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a graph. In this definition, the the nodes n_a and n_b are considered to be connected if $(n_a, n_b) \in \mathcal{E}$ or $(n_b, n_a) \in \mathcal{E}$ (or both).

Directed and Undirected Graphs In some situations, the relation expressed by a graph is symmetric in a sense that both connected nodes are equally part of the relation (e.g. when modeling similarity of entities). In other cases, the relations may have two different roles (e.g. specialization of entities: a square is a kind of rectangle, but not vice versa). A graph is referred to as *undirected*, if the following condition holds:

$$(n_a, n_b) \in \mathcal{E} \iff (n_b, n_a) \in \mathcal{E} \tag{2.1}$$

A graph is referred to as *directed*, if the following condition holds:

$$(n_a, n_b) \in \mathcal{E} \iff (n_b, n_a) \notin \mathcal{E} \tag{2.2}$$

Labeled Nodes and Edges In a graph, each node and each label is considered to be unique. Nevertheless, sometimes nodes and edges need to be equipped with additional properties assigned to them (some of them may be identic for multiple nodes or edges). This is realized by labeling functions. One labeling function $l_{\mathcal{N}} \colon \mathcal{N} \to \mathcal{L}_{\mathcal{N}}$ is used to assign nodes label from a labeling set $\mathcal{L}_{\mathcal{N}}$. Elements of such a set may again be sets, so that a node can have multiple properties. Likewise, there is a labeling function $l_{\mathcal{E}} \colon \mathcal{N} \times \mathcal{N} \to \mathcal{L}_{\mathcal{E}}$ used for assigning edges different properties.

2.1.2. Graph Matching

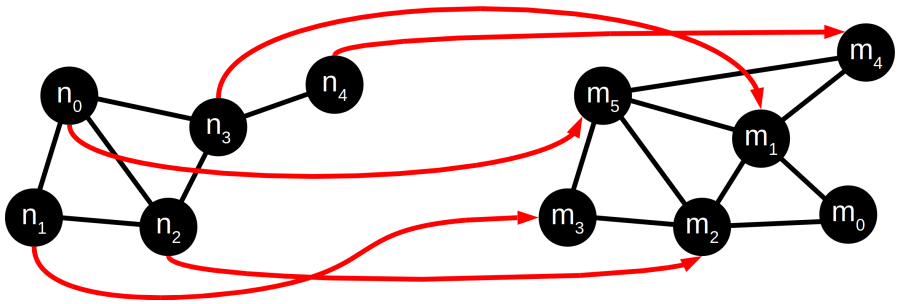


Figure 2.2.: Illustration of Subgraph Matching of Unlabeled Graphs. The graph on the left-hand side is a subgraph of the one it the right-hand side.

Graph matching deals with the generation of mappings between two graphs (see figure 2.2). Formally, in the course of the thesis at hand, a subgraph matching algorithm is an algorithm that creates injective maps $m \colon \mathcal{N}_\alpha \to \mathcal{N}_\beta$ from the nodes of graph α to the nodes of a graph β. The map has to preserve the graph structure of α in β:

$$(n_a, n_b) \in \mathcal{E}_\alpha \wedge m(n_a) = p_x \wedge m(n_b) = p_y \implies (p_x, p_y) \in \mathcal{E}_\beta \tag{2.3}$$

Additionally, a subgraph matching may enforce same labeling of nodes and edges:

$$l_{\mathcal{N}_\alpha}(n_a) = \aleph \wedge m(n_a) = p_x \implies l_{\mathcal{N}_\beta}(p_x) = \aleph \tag{2.4}$$

$$l_{\mathcal{E}_\alpha}(n_a, n_b) = \aleph \wedge m(n_a) = p_x \wedge m(n_b) = p_y \implies l_{\mathcal{E}_\beta}(p_x, p_y) = \aleph \tag{2.5}$$

VF2 and index-based methods exist as optimized implementations of such subgraph matching methods.

2.1.3. Graph Databases

Graph Databases[13] store information and model the relations of different entities on the basis of graphs. Neo4j [38] is an open-source implementation of such a database concept. In order to access and manipulate graphs in Neo4j, a dedicated query language called *cypher*. Cypher itself borrows many concepts and syntax elements from SQL, but specializes in the querying and manipulation of graphs. In the course of this thesis, only the querying features of cypher are utilized.

The syntax and semantics of cypher are illustrated by an example. Consider a database that models tools for craftsmen. The following query can be used to learn all names of the chemical elements that are contained in the nails that a hammer may treat:

Listing 2.1: Sample Cypher Query

```
MATCH (a:‘hammer‘)-[e1:‘treats‘]->(b:‘nail‘)-[e2:‘contains‘]->(
    c:chemicalElement) RETURN DISTINCT c.Name;
```

Résumé on Graphs Graphs are a tool for modeling relations. Their properties have been intensively studied in literature and different algorithms (e.g. for graph matching) are available as practical implementations.

2.2. Artificial Neural Networks

Artificial neural networks (ANNs) are a biologically motivated approach in the field of artificial intelligence (AI). There is a wide range of different kinds of ANNs described in literature, some of them are pointed out in more detail below. Generally, ANNs mimic the behavior of neurons of biological organisms. More precisely, the interaction between these units is of most interest here. An ANN usually has a fixed number of inputs and outputs (often described as a vector) and neural units in between. In the neural network, units are connected to each other by so-called *weighted connections*. The weights of these connections influence how the neural units communicate. These weighted connections are initiated with random values when creating the ANN and adapted during a *training* process in order to a imprint a desired behavior to the neural network. After such a training process, the ANN can be deployed in a productive situation, where it is used to *inference* from input. In the following, a trained neural networks is referred to as *model*.

2.2.1. The Perceptron as an Analogon to the Neuron

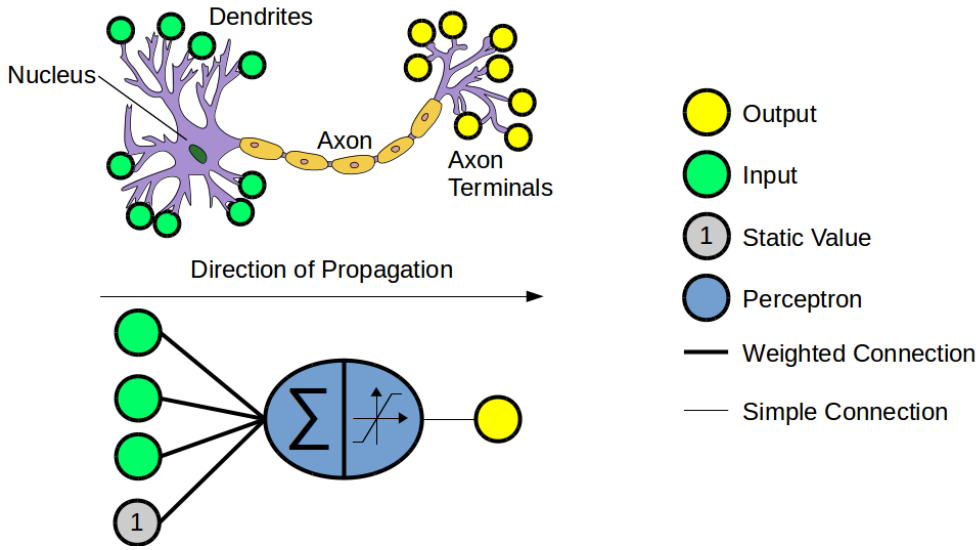


Figure 2.3.: Comparison of a Simplified Neural Cell to the Concept of a Perceptron. Contains material from [9].

Neurons are information processing cells that exist in a wide range of biological organisms. Neurons possess several projections that can roughly be divided into axons and dendrites. Both connect a neuron to other neurons. While dendrites mainly carry the receiver devices of a neuron, axons propagate signals and the axon terminals transmit the cell state signal to other neurons. These inter-cell connections are referred to as synapses. Roughly speaking, the cell state can be described as either activated or inactivated. Whether or not the cell is activated is determined by the incoming signals, where each incoming signal can contribute either to raising the likelihood of an active cell state (activation) or lowering this probability (inhibition).

Perceptrons mimic the information processing capabilities of neurons (see fig.2.3). The output of each perceptron is calculated by an activation function applied to the weighted sum of inputs (r is the number of input connections to the perceptron):

$$a = f_{activation}(\sum_{i=0}^{r} w_i x_i) \tag{2.6}$$

Usually a static 1 signal is added to the set of inputs. Doing so enables to output a 1 as in output in cases where a 1 is desired as an output despite all inputs are 0.

There are several ways to choose the activation function. Generally, the functions should be limited to either $[0, 1]$ or at least $[-1, 1]$ and saturate for $x < -1$ and $x > 1$ to mimic the idea of a neuron with an inactive and an active state. As pointed out below, the function also has to be differentiable. So the most common activation functions are:

$$f_{activation}(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

and

$$f_{activation}(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.8}$$

These two activation functions differ mainly in their output range. While 2.7 has an output interval of $[0, 1]$, the function 2.8 has an output range of $[-1, 1]$. Which function is used mainly depends on the application, both can be used for training. Activation functions are usually monotonic and have a threshold, at which they transition from one state to the other. This threshold is implemented by the weighted connection from the static 1 to the summation of the perceptron.

2.2.2. Multi Layer Perceptron

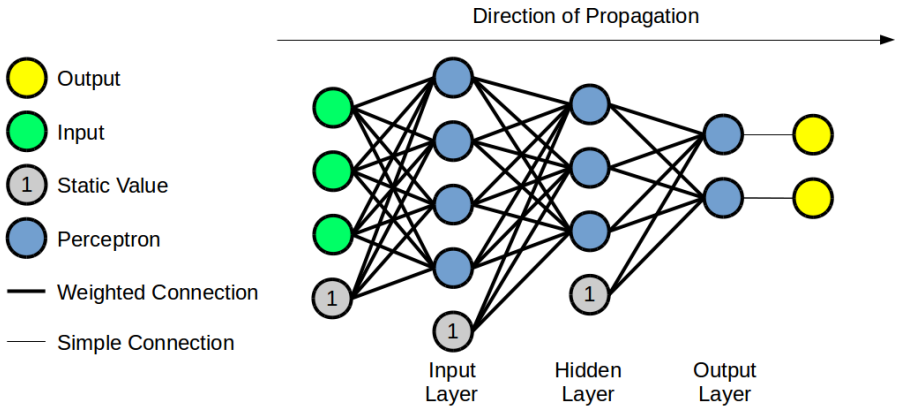


Figure 2.4.: The Multi Layer Perceptron. The input (green) is propagated from left to right. Apart from connections to the preceding layer, every perceptron has a connection to a static unit, serving as adaptable threshold.

The multi layer perceptron (MLP) consists of a stack of layers of perceptron units, in which the outputs of every layer are fed into the subsequent layer (see fig. 2.4). Hence, there is a full interconnection between the preceptron units of each layer, while the perceptron units of each layer are not connected to each other. The number perceptrons in each layer as well as the number of layers themselves are arbitrary, and appropriate values for these so-called hyperparameters are usually determined by trial-and-error experiments. Likewise, the number of MLP net inputs and output perceptrons is usually determined by the application.

Every connection is equipped with a so-called weight determining the influence an outgoing perceptron has to the ingoing. Since two consecutive are assumed to be fully connected, their connections are usually described by a weight matrix. The inputs of the MLP are usually expected to be ranging between zero and one.

Backpropagation The weights of an ANN are usually initialized with random values. During the training process, these weights are altered so that the forward function of the ANN approaches the desired function. One commonly used algorithm to accomplish training is the so-called backpropagation [29] (BP) algorithm. At this point, BP training is shown for MLPs.

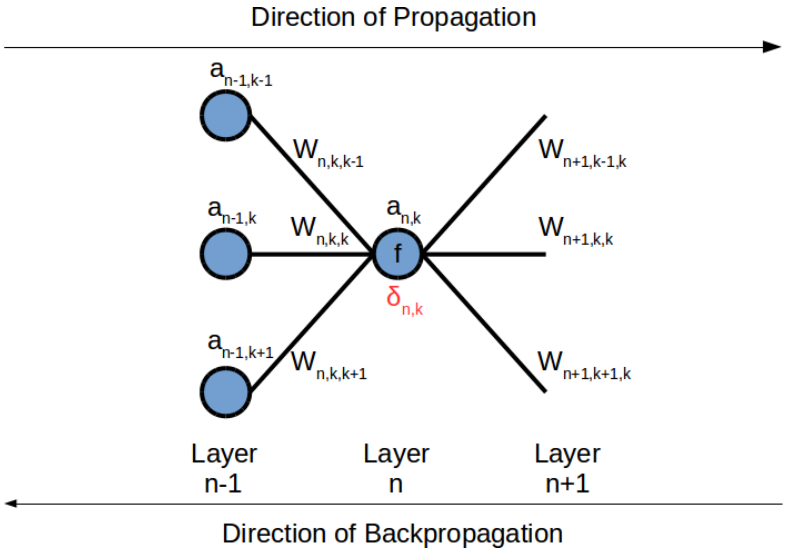


Figure 2.5.: Direction of Propagation and Backpropagation.

In order to apply BP, an input is propagated forward through the MLP (see figure 2.5). During forward propagation, for every perceptron inside the MLP the activation a and derivative d of the activation function at the same argument are calculated ($a_{n,k}$ is the activation of the k-th perceptron in the n-th layer, K_n is the number of perceptrons in layer n):

$$a_{n,k} = f_{activation}(\sum_{i=0}^{K_{n-1}} w_{n,k,i} \cdot a_{n-1,i}) \tag{2.9}$$

$$d_{n,k} = f'_{activation}(\sum_{i=0}^{K_{n-1}} w_{n,k,i} \cdot a_{n-1,i}) \tag{2.10}$$

The BP starts at the last layer, where its error is calculated from the difference of the layer's output and the target values that are known from the ground truth. The error is then propagated back through the layers by the following recursive equation:

$$\delta_{n,k} = d_{n,k} \cdot \sum_{i=0}^{K_{n+1}} w_{n+1,i,k} \cdot \delta_{n+1,i} \tag{2.11}$$

The error $\delta_{n,k}$ is now used to calculate the weight update $\Delta w_{n,k,i}$, which is added to the existing weights:

$$\Delta w_{n,k,i} = -\mu \cdot \delta_{n,k} \cdot a_{n-1,i} \tag{2.12}$$

2.2.3. Recurrent Neural Networks

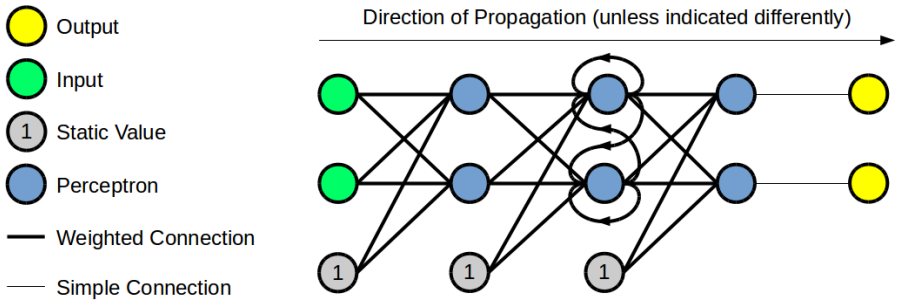


Figure 2.6.: A Recurrent Neural Network. The second layer is equipped with a recurrent connection. In each time step, the recurrent connections feed the outputs of the previous time step into the recurrent layer.

MLPs always map an input vector of fixed length to an output vector of fixed length. That makes is difficult to handle input of varying size. To overcome this problem, recurrent neural networks have been introduced that are scanning over an input of variable size in a series or time steps. In order to incorporate knowledge from previous time steps, at least one hidden layer needs interconnections to it previous activation (see figure 2.6).

Backpropagation Through Time So far, the BP algorithm has been introduced for the training of MLPs. In order to apply BP to RNNs, the recursive connections of the RNN have to be resolved in a way that the RNN appears as a normal MLP to the BP algorithm. For that purpose, the inputs of the recursive connections at a certain time step t are replaced by copies of the recursive connection's outputs one time step back in time. This process (also referred to as *unfolding in time*, see figure 2.7) is iterated from time step t back to the first time step. The BP algorithm can now be applied as if the RNN was an MLP. This proceeding is referred to as backpropagration trough time [28] (BPTT). In order to train an entire sequence, BPTT is repeated for all time steps.

2.2.4. Long Short-Term Memory

One of the main motivations for the development of LSTM[31] was the discovery of the vanishing gradient problem 2.8: Consider an RNN has to store a value over a great amount of time steps. Before training, all weights are initialized with random values. During training, a weight may approach 1, but even a small derivation from 1 results in a vanishing of the stored value during propagation. Likewise, when applying backpropagation in situations with many time steps, the error signal decays over time [30]. More

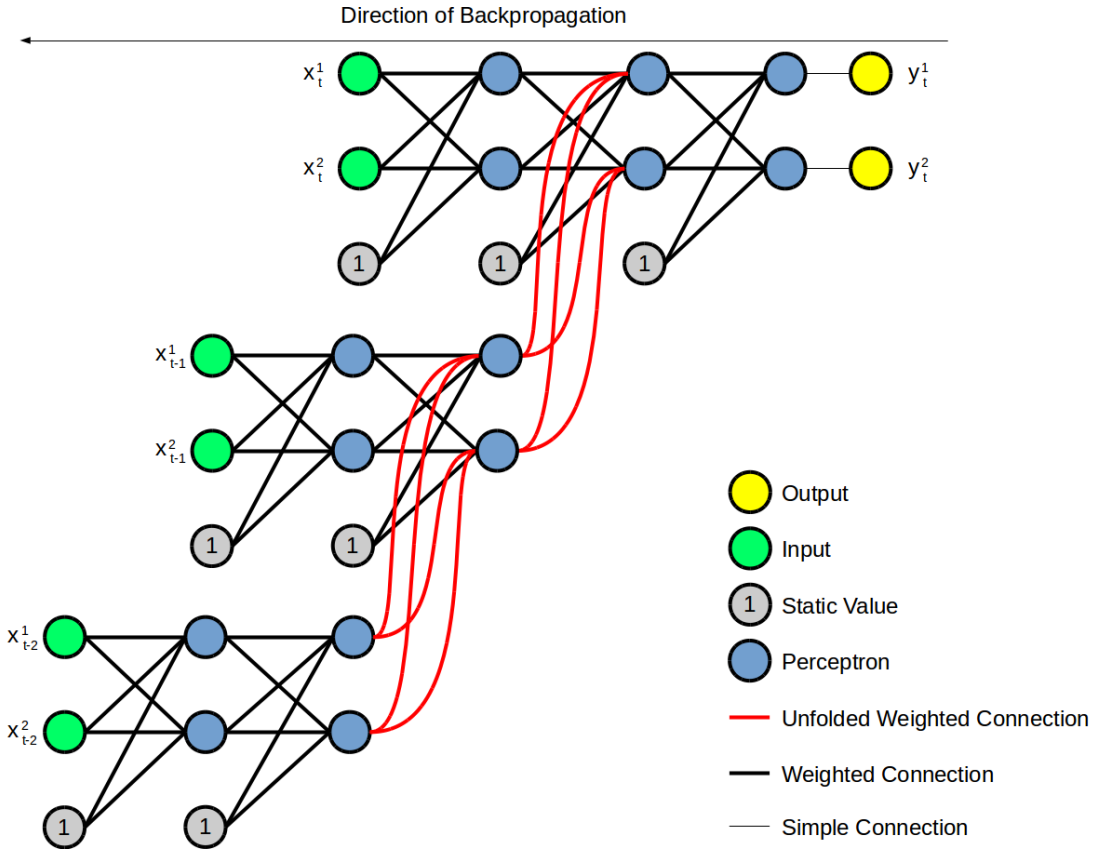


Figure 2.7.: Unfolding an RNN in Time. Doing so allows for applying the standard BP algorithm.

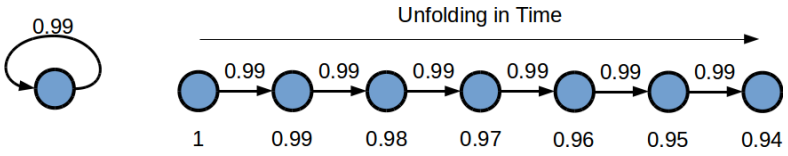


Figure 2.8.: Illustration of the Vanishing Gradient Problem.

precisely, by equation 2.11 the error signal at the perceptron at a certain layer is usually smaller than the error signal at the perceptron in the next layer. Hence, only small error signals are present in the first layers which makes them harder to train. Likewise, when using BPTT while dealing with large sequences the error signals in the first time steps are hard to train.

In this thesis, the LSTM implementation of the ocrolib [19] is used (see figures 2.10 and 2.9 for an overview). This implementation makes use of the two activation functions introduced above:

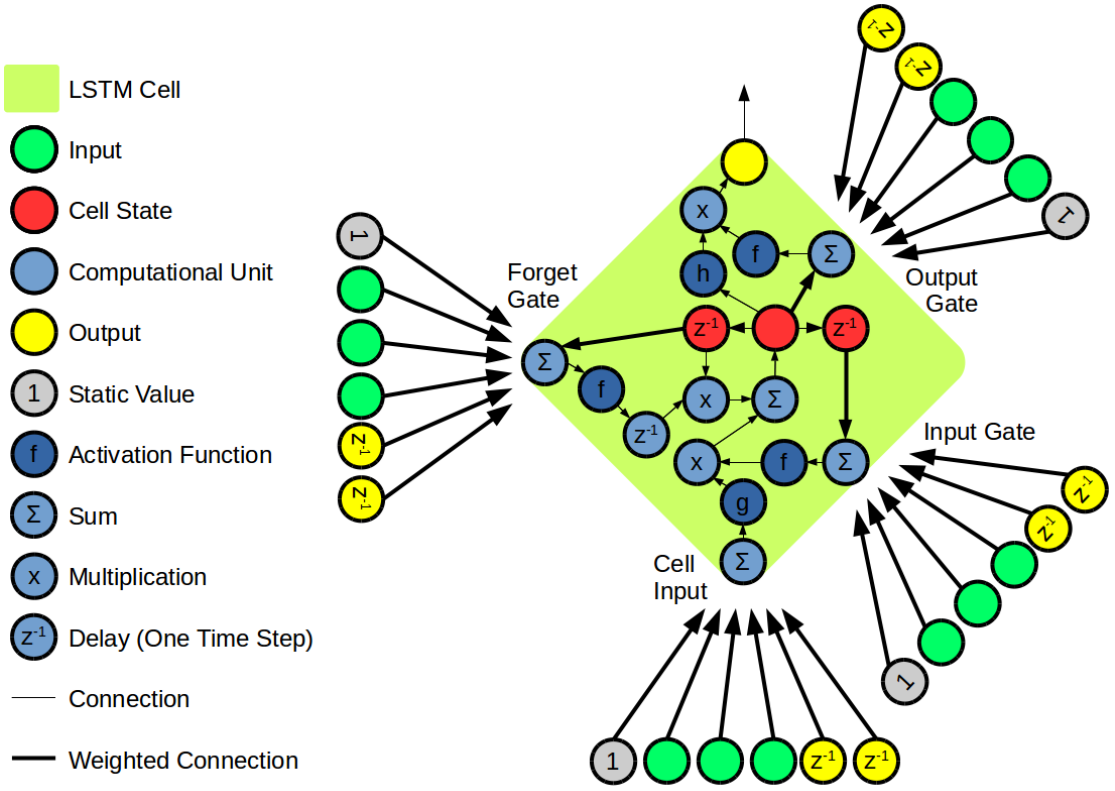


Figure 2.9.: Long-Short Term Memory Cell as Implemented in OCRolib. The initialization constraints and the cell state recursion are omitted. The Inputs passed to each gate of the LSTM cell consists of a static one value, the actual inputs passed to the LSTM layer as well as the outputs of all LSTM cells in the LSTM layer from the previous time step. The figure makes use of the Z-transform notation [44] (z^{-1}) used in signal processing to indicate time delays.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.13}$$

and

$$g(x) = h(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.14}$$

The key idea of the LSTM is to provide a connection with a fixed weight equal to 1 within the recursion in order to allow for storing values for an unlimited amount of time steps within the cell state. Cell states are one central point of recursion, but the cell output also has a recursion. The cell state controls the output, but also the input is taken into account here. The cell state may be influenced by the input gate and the cell

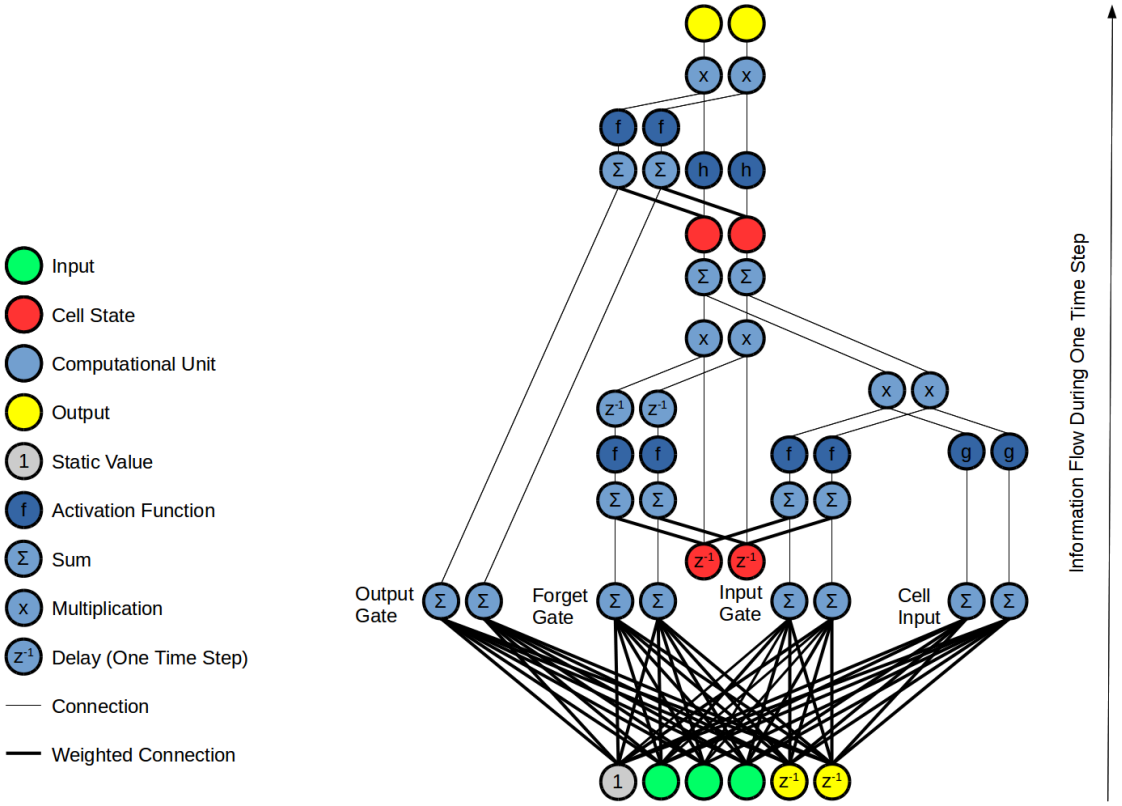


Figure 2.10.: Long-Short Term Memory with 2 cells and 3 inputs as implemented by OCRolib. The initialization constraints and the cell state recursion are omitted.

input. Cell input and input gate mainly differ from each other in the fact that the cell state also controls the input gate while it doesn't control the cell input. The forget gate however was introduced after the original LSTM was released. Its purpose is to reset the cell state from time to time, avoiding an unbound growing of the cell state values [25]. For that purpose, the forget gate can erase the previous cell state completely (simply because its activation is a factor with which the cell state is multiplied).

As a consequence from the structure, the number of LSTM cells and inputs are independent, but the number of cells determines the number of outputs of the LSTM layer. Consequently, in order to have a number of outputs different from the number of LSTM cells, an additional layer is required; for example a perceptron layer or an MLP.

2.2.5. Bidirectional RNN

In its simplest form, an RNN scans a the input data in one direction (most intuitively forwards in time or along an space-related axis). In the following, the computation steps are referred to as time steps, although the scanned data are not necessarily related to a physical time domain (e.g. when considering a image data). Consequently, at each

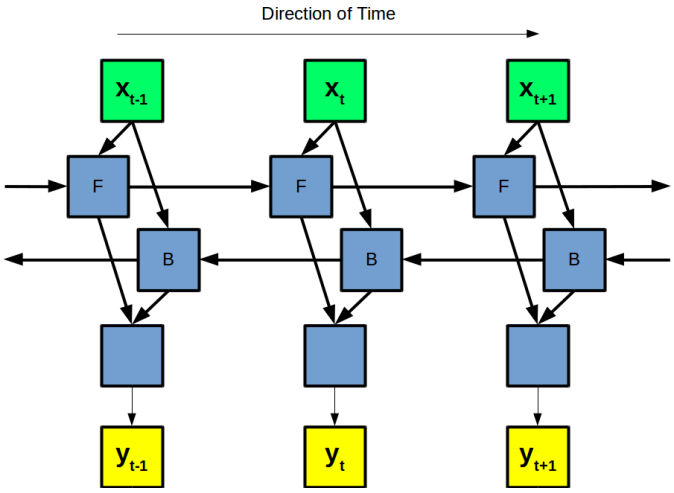


Figure 2.11.: Bidirectional Recurrent Network. The BiDiRNN consists of a forward network (F), a backward Forward (B) and a unifying network (below F and B). This figure is an adaptation from [49].

time step, only information up to that time step is available to the network. For some applications, information from future time steps would be helpful to solve a problem. As a makeshift, the target value of the RNN could be shifted back in time by n time steps, resulting in the availability of the n future time steps in every point of time. As pointed out in [49], this strategy only works for a limited amount of time steps. In order to solve this problem (and hence allow for an availability of the input information of all time steps in all computation steps) the bidirectional recurrent network structure has been proposed (see figure 2.11) for an overview). In this structure, there is an RNN for each direction of time. The outputs of both RNN have to be merged by a third neural network, for example an MLP (one layer is usually sufficient).

2.2.6. Sequence Generation with ANN

Neural networks can be used to generate sequences. One approach to accomplish this goal is to train a neural network to predict the next element of a sequence. Such a predictor is trained of a set of existing sequences, always aiming to compute the nth element of a sequence given the fist $n-1$ elements of that sequence. Now, the trick is to iteratively predict an element and then feed the predicted element back into the model's input [27].

Résumé on Neural Networks Neural networks are a powerful method for automatically generating systems that solve a given problem. One of their most significant feature is to automatically recognize relations and therefore to be applicable without the need to externally incorporate knowledge about the problem to be solved. However, they often need to be provided with a database of pairs with known input and output in order

acquire the intended behavior.

2.3. Human Computer Interaction

Human computer interaction (HCI) deals with the design and analysis of computer interfaces that are handled by human users. HCI focuses on the needs of users, hence the user interface (UI) should be designed in a way that it respects the needs of the user. The International Organization for Standardization (ISO, [4]) deals with different aspects of HCI in its norm 9241. HCI is a interdisciplinary research field that touches many other fields like psychology, ergonomics and computer graphics.

2.3.1. Usability

One of the main goals of HCI is to provide tools that enables the designer to maximize the *usability* of a UI. There are multiple definitions of usability in literature that incorporate different aspects (or so-called dimensions) that describe usability. Different dimensions are considered to be decoupled [24], despite the fact that measures that are intended to affect one dimension also might affect others.

2.3.1.1. Dimensions

Despite different definitions of usability exist (e.g. [50],[41]), this thesis uses the definition of usability by the International Organization for Standardization. The ISO standard 9241-11 defines usability under the following dimensions [33]:

- Effectiveness: This term describes how well a user can perform a certain task at all using the UI. If the UI lacks effectiveness, the user is unable to accomplish tasks using the UI.
- Efficiency: This term described how efficient a user may operate the system. Efficiency is related to resources, often time is the main resource of interest.
- User's Satisfaction: This term simply describes how satisfied a user is when using the system.

2.3.1.2. Measurements

In order to assess the usability of a software, measurements and metrics for each dimension have to be defined. Usability is often measured in the course of a dedicated experiment in which test users are executing a previously defined task on the examined UI while their behavior is captured by several means. Often participants of such experiments are also interviewed after the actual experiment (e.g. a questionnaire is used). In such a setting, effectiveness might be assessed by measuring the degree to which a user completes the well defined sample task. Efficiency might be captured by rather trivial mathematics and observations (e.g. time measurements to complete a task or counting

of clicks the user made to complete the task). The user's satisfaction however might be harder to measure. Questionnaires may serve as an important information source here.

2.3.2. Human-Centered Design

The term usability is part of the human-centered design approach [34], which aims to systematically create UIs that fits the needs of the human end-user. Human-centered design has been normed under the ISO standard 9241-210 (formerly ISO 13407). Human-centered design involves an iterative design process, an early involvement of the end-users as well as other parties involved in the product's lifecycle and a focus on the needs, abilities and tasks of the user [33],[26].

Apart from that, descriptive guidelines for developing UIs exist, e.g. [51]. In fact, many facets of user-centered design are heuristics that try to prevent the developers of a UI to make certain mistakes (or correct them) rather than offering prescriptive rules to generate a UI given a certain problem. For example, cognitive walkthroughs and heuristic evaluations rather try to organize the evaluation process. Likewise, experiments with users try to spot problems with a UI prototype.

2.3.3. Intelligent User Interfaces

In many software systems that employ artificial intelligence, the AI part is relatively separated from the user interface. The field of intelligent user interfaces (IUI) deals with the design of software systems, in which UI and AI are tightly coupled.

Résumé on HCI HCI is a discipline that deals with the constructions of appropriate user interfaces for human users. Different theoretical frameworks for describing user interaction and designing interfaced have been proposed, but many of them remain rather descriptive. Hence, design rules rather guide the developer during the design process instead of proposing specific solutions for the general design problem. Nevertheless, these tools can be useful in the course of an user interface design to avoid known problems.

3. Related Work

The question, to what extend architects can be supported in their creative work during early design phases by algorithms and therefore by software solutions has already been investigated in literature. From the *architectural theory* point of view, the matter to what degree creative work can be conducted (or at least supported) by the means of formal description has been investigated. From the *computer science* point of view, different software strategies for supporting architects in certain aspects of design have been proposed. However, most of these approaches only cover some of the problems an architect encounters during early design phases. This chapter reviews the existing work similar to the approaches made in this thesis. Likewise, complementary views to the fields touched by the thesis at hand are given.

3.1. Floor Plan Pattern Generation

Or et al. [43] focuses on the understanding of 2d floor plan images and the subsequent 3d model generation from it. This work could be used in future to help populate a dedicated database.

Fernandez [22] deals with the general problem of automated object design related to the spatial allocation problem, and touches the application of floor plan design.

Lee [36] investigates how MLPs can be used to recreate the design process of a certain style, more precisely the Neo-Plasticims by Piet Mondrian. More precisely, the author trained an MLP to generate finished pictures of that style given given raw sketches. Finally, the author points out the influence of Mondrian to architects and applies floor plan sketches to the MLP model to illustrate structural similarities. As one limitation of MLPs, both the sketches and the final output images have to be of a fixed size. Likewise, the resulting MLP model is only capable of generating images of the mentioned style. Nevertheless, the author illustrated how a chain of rather complicated design steps can be performed by ANNs in principle.

Afonso et al. [11] investigate how insights from system theory (more precisely phase space analysis) can be used for the generation of residential buildings. The rationale from the system theory used is the fact that in certain phase spaces between chaotic regions and periodic regions there is a region of self-organization. This region can be employed for purposes of architectural pattern generation. The authors emphasize that system theory provides a new way, architecture can be contemplated:

> *"Design can be inspired by this random nature, as nature is not designed with strict parameters, but it may evolve from a core of generative expressions of code. Architecture in fact could be conceived in a similar way the generative randomness develops into complex organisms, from the very bottom processes. This particular organizing principle - from code to shape - in Nature systems, acts along this paper only as an inspiration to the design process."* [11]

Functionally, the presented work generates 3D structures given a set of requirements. In the course of the work, a kind of controlled randomness is employed. Methodically, they used the following algorithm: given a set of (two) previously generated units and an algorithm for joining two units geometrically, they constructed an evolutionary algorithm. As a fitness function, they simply asked a human user so select favorable settings. Thereby, the user decides about which of the generated joinings should procreate. For generating geometric assemblies of connected units to let the user from, they employed randomness for determining the joining parameters. This process is iterated over multiple steps so that the resulting structures grow over time.

As one big advantage, the algorithms puts the user in the loop, hence give him control the process. However, the obtained results appear to be somewhat artificial and raises the question of actual applicability.

Alexander [12] generally tries to describe the process of design. The author makes use of graphs to model restricting agents occurring in a design process (e.g. economics in the production process, overall product performance, environmental impacts of the product or different user needs) and the relations between them. The author emphasizes that even in simple products these graphs and the related problem solving are of enormous complexity since each of these agents is related to a great amount of other agents. The author concludes that since the complexity has increased over the time, single human beings are no longer capable of solving problems nowadays:

> *"The intuitive resolution of contemporary design problems simply lies beyond a single individual integrative grasp."* [12]

The author considers the presented method rather a visualization tool for the complexity of a design problem than an actual tool for solving it. However, due to the stated complexity of design, the presented methods based on logic and mathematics offer a reasonable tool to cope with this issue in the author's opinion. Nevertheless, the author argues that the introduction of a formal, graph-based abstraction level ensures the avoidance of biases caused by language and experience.

Merrell et al. [37] present an end-to-end approach for generating floor plans and 3D models of residential buildings by using bayesian networks [42]. In their 3D visualizations, they allow for different style flavors. This work mimics the existing work flow of an architect: starting with a high level description of the required rooms, creating a connection graph based on that, generate the room layouts and finally a 3D model.

The general workflow of this system (as well as the modeling of rooms) appears to be appropriate to solve the specified problem. However, the work itself is restricted to residential buildings and the bayesian networks approach requires manual modeling that (at least partly) has to be redone when the approach should be applied to another type of building.

Richter [46] deals with the applicability of case-based reasoning (CBR) to design in architecture. The author assumes that the case-based reasoning technology is generally applicable to the problem, but none of the existing approaches in the past has actually been successful regarding widespread, productive use. The author points out that a lack of a universal understanding of the CBR technology and the neglection of theoretical constraints as well as the oversimplification of the problem contribute to failures of those attempts. Generally, [46] considers itself also as a guideline for upcoming CBR implementations and provides concepts and additional constraints for applying CBR to architecture as well as intensive theoretical investigation of this approach.

3.2. Architectural User Interfaces

In late design phases, computer aided design (CAD) software systems are already in widespread use.

Huang et al. [32] focus on helping the user to optimize the energy consumption of a building.

Bhavnani et al. [17] analyzes real world interaction between human users and CAD software and the efficiency of this. The authors state that this interaction is often significantly below maximum efficiency and remark that some users tend to replicate drawing procedures from the analogue world. As one reason for such a behavior the authors analyze manuals of CAD software and find that these books tend to explain individual commands of the CAD software rather in detail than delivering a complete picture of an efficient workflow. Nevertheless, the quality of the end product is not necessarily affected by that suboptimal behavior. The authors therefore argue that the relevant knowledge needed for an efficient workflow should be conveyed explicitly to the user and that automated feedback messages should be employed to hint the user to his suboptimal behavior. The authors finally conclude that optimizing CAD systems needs special attention:

> *"The CAD productivity problem, as we have demonstrated, has to do with deeper mechanisms that can plague the proper use of any new technology or medium. If the CAD productivity phenomena is ignored or explained away by the nature of what CAD systems do, then we are doomed to repeat their mistakes. If, on the other hand, we understand that a new technology often requires reformulating old tasks, then we can spend more time in making that knowledge explicit and minimally disruptive."* [17]

Moelle [40] investigates how architects could use software systems in early design phases and analyzes the weaknesses of existing CAD systems for the architectural domain and outlines different approaches for early design support. Regarding the acceptance of such system, the following is emphasized:

> *"The author comes to the conclusion that improving interaction with the CAAD-system is crucial, in order for them to be used in the design process. [...] Only if the human user interface can be handled truely intuitively, will building models be accepted by designers in the early phases of design."* [40]

4. Methodology

This chapter outlines the concepts that are employed in the proposed solution of the thesis at hand to solve the problems described above in a high-level fashion. Each subsection here describes the concepts relevant to different parts of the proposed solution. Firstly, section 4.1 describes concepts of the user interface (UI). Secondly, section 4.2 explains the concepts of the floor plan search engines. Finally, section 4.3 describes the functional principles of the design proposal functionality.

4.1. Room-Based Digital Conceptualization

As a basic functionality for an architectural design-supporting user interface, a function for entering floor plan concept is required. In order to follow the room schedule working method, the editor tool must primarily be an editor for individual rooms. Hence, rooms are the objects to be manipulated. In contrast to space-oriented editors, walls are always considered a part of a single room only. If a room is moved, all surrounding walls move accordingly.

4.1.1. Abstractness of Rooms

In order to support the user during the entire conceptualization process, floor plans have to be entered in a varying degree of abstractness. Currently, two main degrees of abstractness are defined:

1. Abstract (or Bubble) Mode. The room is simply characterized by its distinct existence.

2. Specific Wall Geometry (or concrete) Mode. The room is characterized by a polygon describing its surrounding walls.

Both modes have to visualized in distinct manners in order to avoid confusion. As a general strategy, abstract rooms are displayed as circular geometric objects (therefore they are also referred to as bubbles) while specific rooms always have a polygonal shape (where no arcs are allowed). To emphasize the difference even more, the surrounding edges should have different styles. However, these two levels allow for further refinements. In the bubble mode, already a function and a size can be assigned. Likewise, the property of whether or not the room has a window can be assigned. Finally, the room can be connected to other rooms. When switching to the specific wall geometry mode, all these properties should be adopted as they stand for convenience. As a first problem, the size usually changes after the user defined a wall polygon. This issue is mitigated by

the fact that the size of the polygon can be considered a more precise approximation of the user's wish regarding the room's area size. Having a specific wall geometry, further refinements are possible. The links of a room to other rooms can be refined by assigning holes in the wall that specify these connections.

4.1.2. Abstractness of Links

The requirement for supporting a varying degree of abstractness imposes several problems: Connections between rooms of different abstractness levels need to be supported. In fact, even two rooms with different levels of abstractness should be supported for maximum flexibility. Consequently, connections also bear different levels of abstractness. In the course of this thesis, three levels of abstractness have been considered theoretically:

1. A room is connected
2. A room is connected via a specified wall
3. A Room is connected via a specified 'hole' in a wall

While these different abstractions appear to be reasonable in terms of flexibility, offering all these possibilities can cause the user to be confused. Since the second option appeared to be rather neglectable, only the first and the last are kept. Apart from that, specific rooms can be connected via abstract links. Passages are the default link type.

4.1.3. Physical Dimension of Walls

Following the room-oriented concept, walls are primarily surrounding borders of a room. In order to allow for convenient arrangement of rooms, the rooms should be movable seamlessly around the drawing area. Nevertheless, in the final floor plan, rooms need to be systematically arranged are separated by physical walls. For convenience of the user, specific wall geometry rooms can snap with each other. However, when the two wall polygons snap, the thickness of the wall is implicitly set to zero. This problem generally arises from walls considered a room-related borders only. This problem could be resolved by specifying thicknesses for different walls. However, since the editor only needs to be used for early design phases, the problem is disregarded here, defaulting to the zero-thickness solution.

4.1.4. Links viewed as parts of Rooms

As implied above, connectors are needed as part of individual rooms to make up links. Hence, a link between two rooms is decomposed into a pair of connectors, where one connector belongs to each of the two room. Abstract links (i.e. links consisting of two abstract connectors) may change their type more easily since the user is not bothered with their internal representation. But with specified holes in a wall, the question arises, how the type of link has to be interpreted when different types of connectors are used. To overcome this problem, different connection types are determined by the pair

of connector types that make the link up. As a general principle, different connector types are considered to be of different degree of separation, i.e. a wall separates two rooms more than a passage. Generally, the link types (and likewise the connector types) defined in 4.2.1 are regarded. The Tab. 4.1 defines how link types are derived from pairs of connector types.

Connector Types	**Wall**	**Entrance**	**Door**	**Passage**
Wall	Wall	-	-	-
Entrance	-	Entrance	Entrance	Entrance
Door	-	Entrance	Door	Door
Passage	-	Entrance	Door	Passage

Table 4.1.: Connector types of a link determine its type. Forbidden combinations are indicated by a '-' symbol. For example, a connection that is a hole in the wall of one room (passage) and a door to the other room make up a door connection.

4.2. Semantic Pattern Matching

Floor plans need to be brought in a formal pattern in order to be stored and to be processed. Likewise, metrics for assessing similarity needs to be established in order to search for similar floor plans (retrieval). In the following, this problem is solved by considering a floor plan as a graph. In such a graph, each node is a room of the floor plan. Hence, labels (properties) of a node describe a room's features and the connections between the nodes describe the relation between the corresponding rooms. A first task is to determine alphabets of labels for nodes and edges to describe these properties. In the course of this thesis, this problem is solved in AGraphML.

Graphs allow not only for storing the floor plans in a simple way, but also to apply subgraph matching as a powerful and well researched tool for retrieval purposes. The core idea is that two floor plans are similar if their corresponding graphs match. Nevertheless, this idea has to be defined more precisely in order to be applicable in real-world scenarios. Given a full graph of a floor plan with a variety of node and edge labels, abstractions have to be found that allow for an effective application of subgraph matching. These abstractions are referred to as *semantic fingerprints* in the following. Graph based fingerprints mainly differ in the restriction of labeling of nodes and edges. In any case, they embody knowledge manually defined by experts in the field of architecture about aspects that characterize a floor plan.

4.2.1. AGraphML

The Architectural GraphML (AGraphML) file format has been developed to describe floor plans in a simple manner. AGraphML is a specification of GraphML [18] (which is in turn a specification of XML [52]). AGraphML been developed in the course of

Name	Comment
ROOM	Used if function not further specified or no other function fits.
BATH	
LIVING	Can also be used for dining.
SLEELING	
KITCHEN	
STORAGE	Can also be used for attics and utility rooms.
WORKING	Used for offices and workshops.
PARKING	
CORRIDOR	A room that is connecting rooms. Fallback for stairways.
TOILET	
EXTERIOR	Can be used for balconies or terraces
CHILDREN	

Table 4.2.: Node Types in AGraphML.

Name	Description	Traversable
WALL	Two rooms share one or more walls. Within the connecting wall segments, there are no holes.	No
ENTRANCE	Two rooms are connected by a reinforced door. Usually used to separate apartment units.	YES
DOOR	Two rooms are connected by a simple door.	YES
PASSAGE	Two rooms are connected by a doorless hole in a wall.	YES

Table 4.3.: Edge Types in AGraphML.

a collaboration between TU Munich and DFKI. The first version of AGraphML has been described in [35]. After the first AGraphML standard was developed, it has been extended in the course of this thesis to serve further needs (some attributes of the original specifications are not actively used in the thesis at hand). A sample floor plan with a correllated AGraphML file is shown in fig. A.1 and fig. A.1, respectively.

The key idea of AGraphML is that floor plans are modelled as graphs: each room is represented by a node, a connection between two rooms is represented by an edge. Since GraphML already defines a basic graph structure, the AGraphML specification simply consists of a set of attributes, which a graph, a node and an edge can bear (see Tab. 4.4, Tab. 4.5 and Tab. 4.6). Furthermore, edge types (see Tab. 4.3) and node types (see Tab. 4.2) are defined. For describing the geometric attributes (polygons and points) of rooms, AGraphML makes use of the so-called *well-known text* convention [5] [6].

The choosen format is rather crude. The set of room functions does not cover all types of rooms in real-world scenarios. In the following, the tag `windowExist` is equated with the property of a room to have natural light, although there are cases in which a room with natural light may not have a window (e.g. in the presence of a light shaft) or a

Name	Type	Description	Mandatory
imageUri	String		No
imageMD5	String		No
validatedManually	Boolean		No
floorLevel	Float		No
buildingID	String		No
ifcUri	String		No
bimServerPoid	Long		No
alignmentNorth	Float		No
geoReference	String		No

Table 4.4.: Graph Attributes in AGraphML.

Name	Type	Description	Mandatory
name	String	User-defined Room Name, not to be confused with GraphML's Node id.	NO
roomType	String	One of roomType	NO
center	String	WKT Point	YES
corners	String	WKT Polygon	NO
windowExist	Boolean		YES
enclosedRoom	Boolean		NO
area	Float		YES

Table 4.5.: Node Attributes in AGraphML.

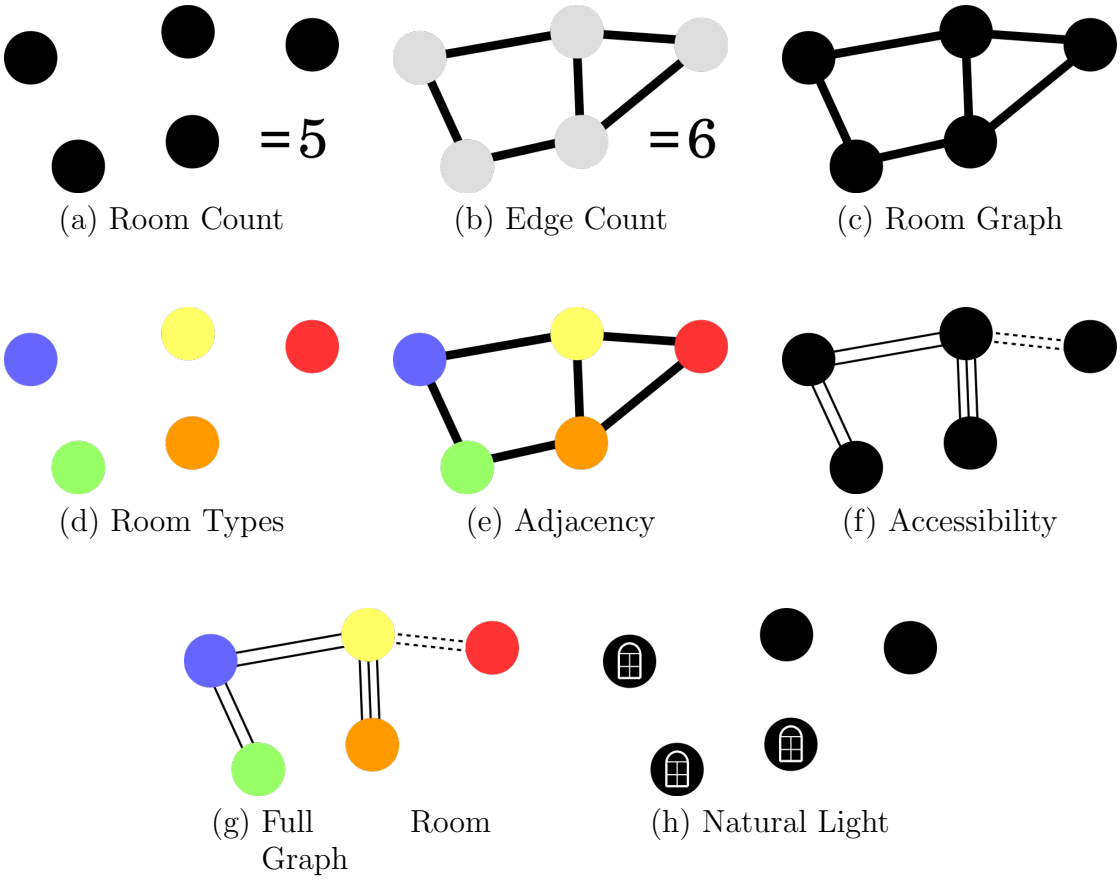
Name	Type	Description	Mandatory
sourceConnector	String	WKT Linestring	NO
targetConnector	String	WKT Linestring	NO
hinge	String	Either `LEFT` or `RIGHT`	NO
edgeType	String	One of edge type	YES

Table 4.6.: Edge Attributes in AGraphML.

room with a window does not have access to natural light (e.g. laboratories).

4.2.2. Semantic Fingerprints

The semantic fingerprints presented here have been developed in the course of the Metis project at DFKI and TU Munich.



(a) Room Count (b) Edge Count (c) Room Graph

(d) Room Types (e) Adjacency (f) Accessibility

(g) Full Room Graph (h) Natural Light

Figure 4.1.: Overview Over the Individual Fingerprints.

The following semantic fingerprints (see also fig. 4.1) have been chosen as a basis for the retrieval systems of the the Archistant framework:

Room Count This fingerprint simply measures the amount of rooms in a floor plan. It is therefore expressible as a positive integer number. Hence, the room count fingerprint of two floor plans are equal if their total room count is equal. The type of rooms, their connections and other properties are disregarded here.

Edge Count The edge count fingerprint behaves like the room count fingerprint except that is measures the total number of connections between the rooms of a floor plan. The edge count is therefore also simply a positive integer number.

Room Graph The room graph fingerprint is a graph with unlabeled nodes and unlabeled edges. For each node in a floor plan, there is one unlabeled node in its room graph fingerprint. for each connection between two rooms, there is one unlabeled edge between their corresponding nodes.

Room Types The room types fingerprint is a multiset of room function labels.

Adjacency The adjacency fingerprint is a graph with labeled nodes, and unlabeled edges. For each room in a floor plan, there is one node in its adjacency fingerprint which bears the same function label as that room. For each connection of rooms in the floor plan, there is one unlabeled edge in the fingerprint.

Accessibility The accessibility fingerprints aims to model how a person can move between the individual rooms of a building. Hence, wall connections are disregarded here and the remaining edge types are expressed explicitly. The accessibility fingerprint is a graph with unlabeled nodes and labeled edges. For each room in a floor plan, there is one unlabeled node in its accessibility fingerprint. For each traversable connection between two rooms, there is a labeled edge between the corresponding nodes bearing the connection's type.

Full Room Graph The full room graph fingerprint is a graph with labeled nodes and labeled edges. For each room in a floor plan, there is a labeled node in the full room fingerprint, bearing the room's function as a label. For each traversable connection between two rooms, there is a labeled edge between the corresponding nodes in the fingerprint.

Natural Light The natural light fingerprint is a multiset of boolean values, where the amount of false entries equals the amount of rooms without natural light and the amount of true entries equals the amount of rooms with natural light.

4.3. Iterative Human-ANN Design Collaboration

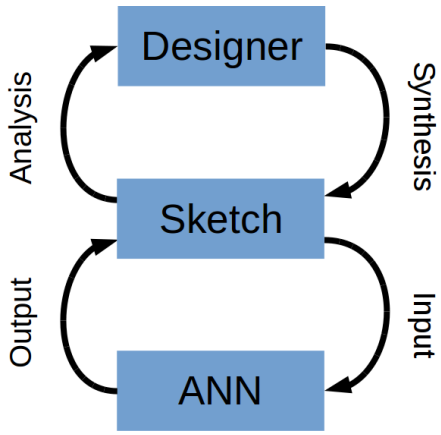


Figure 4.2.: Screenshot of the Archistant WebUI.

Just like the human architect iteratively modifies a sketch to develop a design to solve a problem (as pointed out in chapter 1), an artificial neural network for pattern generation iteratively transformes input to an output, that is consequently appended to the input to serve as a new input (as described in section 2.2.6). The idea is now to couple both processes, resulting in an interactive development process with both an ANN and a human designer in a loop (see fig. 4.2). In such a process, the AI becomes an agent interactively working with the human architect. However, the human user remains the head of the loop by putting him in charge of triggering the AI. The AI itself is an embodiment of human knowledge since the ANN is trained on a database of examples that have been created by human beings. Hence, asking the AI to apply a certain improvement to the equals in asking a couple of human architects to perform the job. Simply put, the AI embodies human knowledge and the human might use the AI as an extension of his own mental capabilities, just like paper has been just for this job in a traditional setting.

4.3.1. Stepwise Recreation of Architect's Behavior

So in order to apply ANNs to the problem of design assistance, an ANN has to be trained on a corpus of existing floor plan designs. More precisely, the floor plans in that corpus have to be decomposed into pairs of input and output in order to train the ANN and later to be recalled into productive use. Generally, a more crude (or abstract) version of a floor plan is transformed in a more sophisticated (specific) one. One way to accomplish this is making use of the the room schedule working method (described in 1.1.1). A floor plan is decomposed into two descriptions of different abstraction levels, where the first is used as input for the ANN and the ladder is used as output. This strategy results in

a model that can carry out a single design step. Consequently, multiple models have to be trained, one for each design step.

4.3.2. From Prediction to Generation

Another strategy is to interpret a floor plan as a single sequence of elements (or actions) that successively make up the floor plan. Such sequences that mimic the architect's behavior can be used to train a sequence predictor. Such a sequence predictor can be used to precisely instruct the AI to predict desired aspects of a floor plan (similar to text completions technologies in office software does).

Generally, an input created by the user serves as initial sequences and is iteratively augmented by predicting single steps and appending them to the existing sequence. Nevertheless, one action might be encoded by multiple input vectors, requiring for a method to abort the process if the desired step is complete. This can be accomplished by dedicated stop symbols. Likewise, it is sometimes requires to lead the predictor in a certain direction. This is accomplished by start symbols, that carry no specific information. In order to save symbols, start and stop symbols can be interleaved.

5. Proposed Solution

This chapter outlines the proposed software solution that was created in the course of this thesis in detail. This software solution implements the concepts described in chapter 4. More precisely, the room-based digital conceptualization (section 4.1) is realized in the web-based sketch editor that is referred to as *WebUI*. The semantic pattern matching (section 4.2) is starting point for the *retrieval systems* that helps the user to get similar floor plan concepts. Finally, iterative Human-ANN design collaboration (section 4.3) is implemented in the *creativity engine*, that generates design suggestions. The software solution proposed here serves as the foundation for the experiments described in chapter 6. This software has also been made public under the name Archistant [2].

5.1. The Archistant Framework

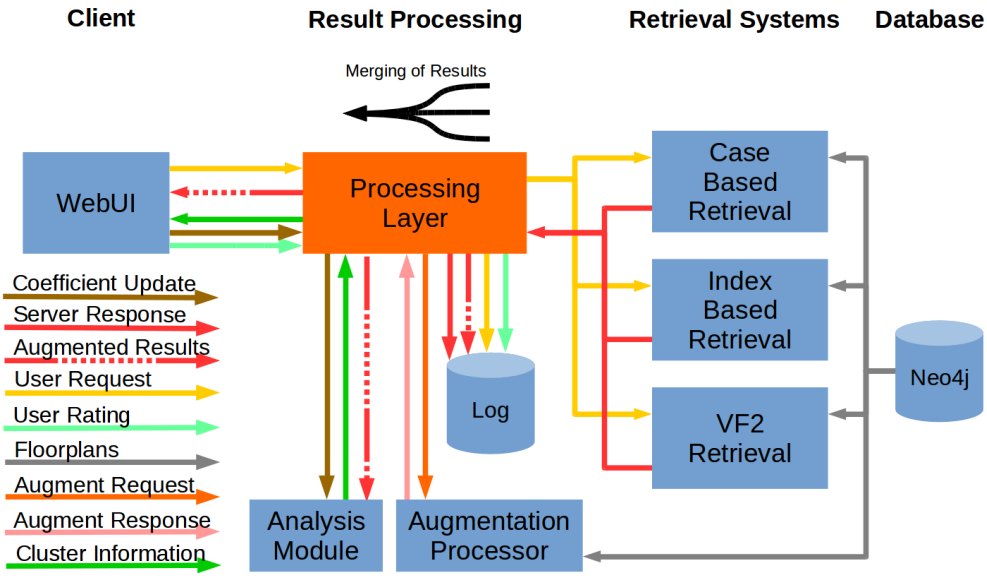


Figure 5.1.: The Archistant Framework (Creativity Engine omitted).

The Archistant framework (see fig. 5.1) consists of a number of different modules, which are connected by WebSocket connections and that exchange floor plan concepts in the form of AGraphML Files.

5.1.1. WebUI

Figure 5.2.: Screenshot of the Archistant WebUI.

The WebUI module (see fig. 5.2 for a screenshot) makes up the interface with which the user communicates to the system. It is the only client-side module and runs inside common web browsers (it is optimized for Firefox [3]). It is written in HTML5/Javascript and makes use of the jquery library [45]. The WebUI consists of the editor (in which the user can create search requests) and the search bar (which can be used to search for similar floor plans and to analyze the results). The Archistant WebUI is explained more detailed below (see 5.2).

The WebUI's task is to generate search queries and creativity requests and provide them to the PL. Likewise, is displays the results of these messages.

5.1.2. Processing Layer

The Processing Layer (PL) is the central hub of Archistant. All communication between the individual modules of Archistant is processed by the PL. However, the Neo4j database is connected to multiple Archistant modules. The PL appears as a WebSocket server to the WebUI and as a WebSocket Client to all other modules. The PL is written in Java8.

The PL allows for multiple connected clients (usually WebUI instances) simultaneously. Likewise, the PL can connect to multiple retrieval systems. The list of systems the PL connects to is given as command line parameters when invoking the PL. Hence, the PL expects these systems to be available when it starts.

5.1.2.1. Performing a Search

When a client (e.g. the WebUI) sends a search request to the PL, the PL forwards this search request to all attached retrieval system. It then waits until all retrieval system sends their result lists. After that, the PL unifies these result lists to a single list (while discards redundancies) and sorts the unified list according to the confidence scores determined by the retrieval systems. As a next step, this unified result list is then handed to both the AP and the AM separately (if one of these systems is not available, the transmission to the regarding module is skipped). If the AP is connected, the PL waits until the augmented list is returned by the AP and then sends this list to the client. Otherwise, the non-augmented list is sent directly to the client. After the (augmented or non-augmented) result list has been sent to the client, the analysis information provided by the AM is send to the client as soon as the AM returns it (but never before the result list has been transmitted to the client).

5.1.2.2. Log Files

The PL logs most of the communication, like search requests, results and user's feedback to the results. These log files are stored in an dedicated XML format (which is simply called logxml). The log files are intended to describe the behavior of the user and the system for analysis and improvement purposes. Consequently, the exact times of user requests and server responses. In general, for each connecting WebUI instance one log file is created and all relevant of the instance are logged into this file. An example for such a file can be found in appendix B.1.

These log files are later used for automatic analysis by the boundary tester module. Apart from that, they could be used in future for automatically assess the quality of the results of the overall system as well as individual retrieval systems. Furthermore, they could be used for automatic improvements of the system (e.g. by automatically adding user's search queries as entries to the database for later retrieval or by applying machine learning to find out what results are most appropriate in a certain situation based on the user's feedback).

5.1.3. Retrieval Systems in General

A floor plan retrieval system in Archistant is a system which - given a query floor plan concept (along with a list of fingerprint weights) - returns a list of floor plans similar to to the query. A retrieval system is directly connected to the PL and appears to it as a server based on the [23] protocol. A retrieval system is provided an AGraphML file that has been generated by the WebUI along with a set of semantic fingerprint weights. Such a search request is syntactically a XML structure. Based on this information, the retrieval system returns a set of hyperlinks to image files, that contain thumbnails representing the results. Each of these thumbnails also has to be referenced by the Neo4j database (as pointed out in subsection 5.1.11).

A fingerprint weight indicated the importance of a fingerpirnt to the user and is implemented as a floating point number ranging from 0.0 to 1.0. A fingerprint that is

assigned a weight greater than 0.5 is considered to be `mandatory`, i.e. such fingerprints must match for a database entry in order to be selected as retrieval result.

5.1.4. Index-Based Retrieval

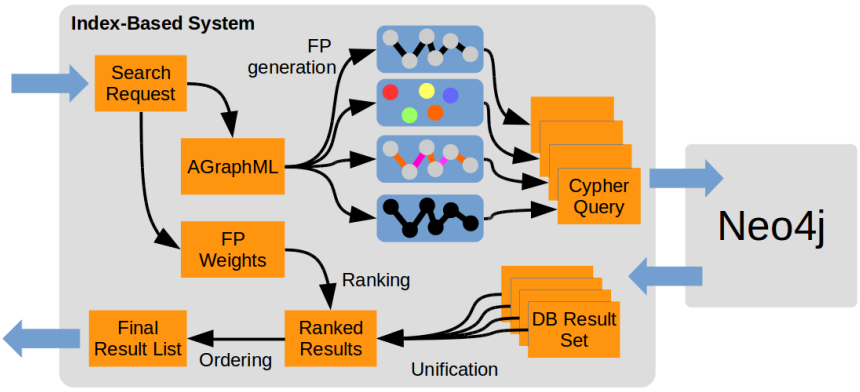


Figure 5.3.: Overview of the Index-Based Retrieval.

The Index-Based Retrieval Module (IB, see 5.3) searches for floor plans similar to a given query by using an so-called index-based matching algorithms. There is a wide range of different index-based techniques available, and the IB effectively uses a Lucene-based index since it queries a Neo4j database, which uses such a indexing. Generally, the IB shows a subgraph matching behavior: a result's fingerprint is considered to match the (graph-based) fingerprint of a query if the graph of the query's fingerprint is a subgraph of the result's fingerprint. Neo4j uses Lucene as indexing technology, what motivated the name of the IB retrieval. The IB is written in Java8.

5.1.4.1. Cypher Query Generation

After the query has been converted from AGraphML to an internal graph structure, cypher queries are generated from this internal structure. By design, the chosen cypher queries effect a subgraph matching behavior.

5.1.4.2. Result Ranking and List Unification

Based on the logic of cypher query matching, a query's fingerprint either matches an database entry's fingerprint or not. Hence, there is no partial matching when considering a single fingerprint. Consequently, the final ranking can be described as:

$$r_{rank} = \sum_{r \in FP} FP_{weight} \tag{5.1}$$

5.1.5. Case-Based Retrieval

The case-based retrieval system has been developed by Viktor Ayzenshtadt. It is explained in detail by [15],[14].

5.1.6. VF2-Based Retrieval

The VF2-based retrieval system has been developed by Qamer Uddin Sabri. It was first described in literature by Sabri et al. [47].

5.1.7. Augmentation Processor

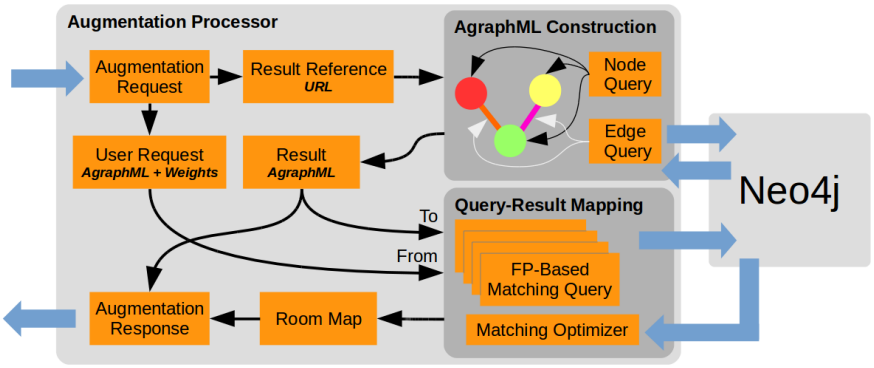


Figure 5.4.: The Augmentation Processor.

The augmentation processor (AP, see fig. 5.4) enriches search results from the retrieval modules by additional information. A result as provided by a retrieval system simply consists of an URL pointing to a thumbnail image. These thumbnail images are uniformly used by all retrieval systems and are referenced by the Neo4j database. Therefore, the AP generates additional information by querying the Neo4j database. There are two kinds of information with which an result is enriched: The result's AGraphML file and a map, which links rooms in the query to rooms in the result. The AP is written in Java8.

5.1.7.1. Generating Result AGraphMLs

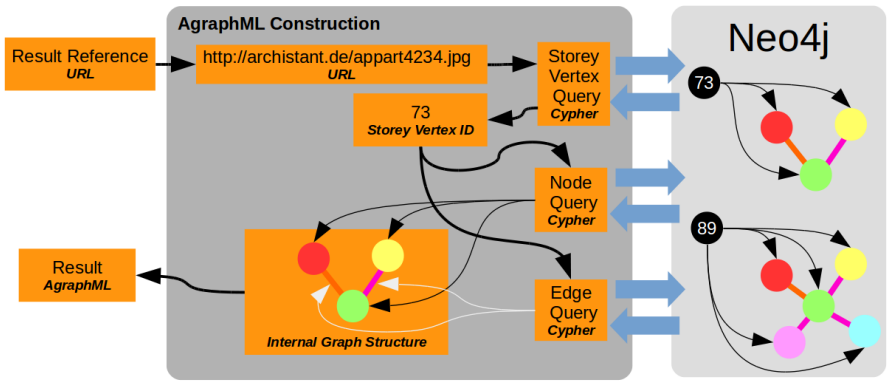


Figure 5.5.: AGraphML Generation in the Augmentation Processor.

AGraphML files are constructed by querying the Neo4j server (see section 5.1.11). This is done in three steps (see fig. 5.5):

Resolving the Storey Vertex First of all, the storey imageURI is used to determine the vertex ID of the database entry that represents the floor plan of interest. For every imageUri referenced by the Neo4j database, there is a `storeyvertex` (node) that organizes the graph structure of the related floor plan.

Retrieving Information about Node Properties Secondly, information about the rooms in the floor plan are retrieved by querying the database for all rooms that are connected via a `stroreyedge` to the `storeyvertex` node with the id of interest.

Retrieving Information about Room Connections Finally, the connections between all nodes that have been found in the last step are retrieved. With all these information present, the final AGraphML file can be constructed.

5.1.7.2. Generating Room Maps

The room map generation utilizes Neo4j's graph matching capabilities.

5.1.8. Analysis Module

The Analysis module (developed by Qamer Uddin Sabri) maps result sets into a two-dimensional space, which axes can be controlled by the user.

5.1.9. Boundary Tester

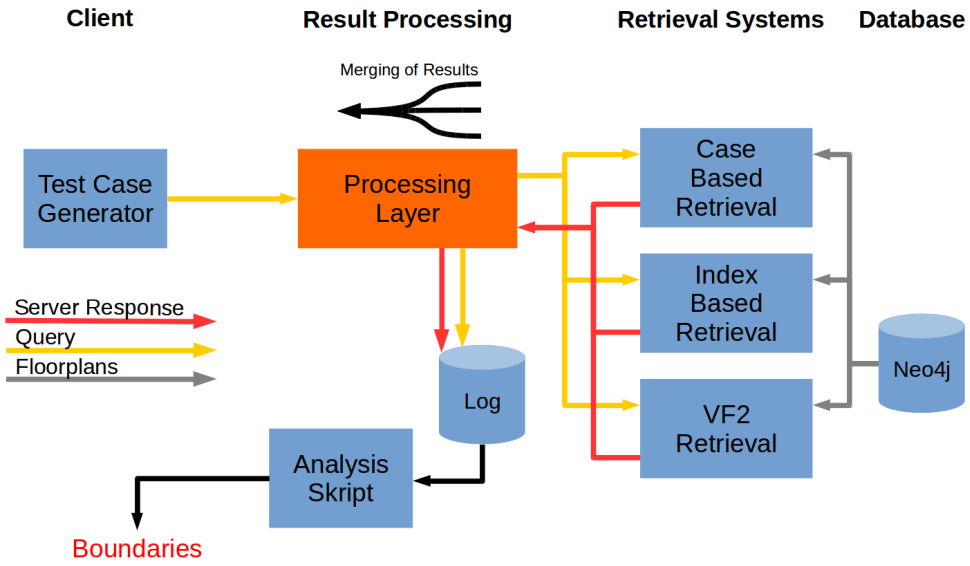


Figure 5.6.: Boundary Tester System Configuration.

The Boundary Tester module (BT, see fig. 5.6) is used to systematically stress-test the capabilities of the retrieval systems. It consists of a module which generates search queries and an evaluation script.

5.1.10. Creativity Engine

The creativity engine (CE) assists the user during conceptualization by generates design suggestions. As a module, it receives creativity request messages from the WebUI that have been conducted by the PL. It is written in Python and makes use of the OCRolib [19]. The CE is explained in more detail in section 5.3.

5.1.11. Neo4j Database

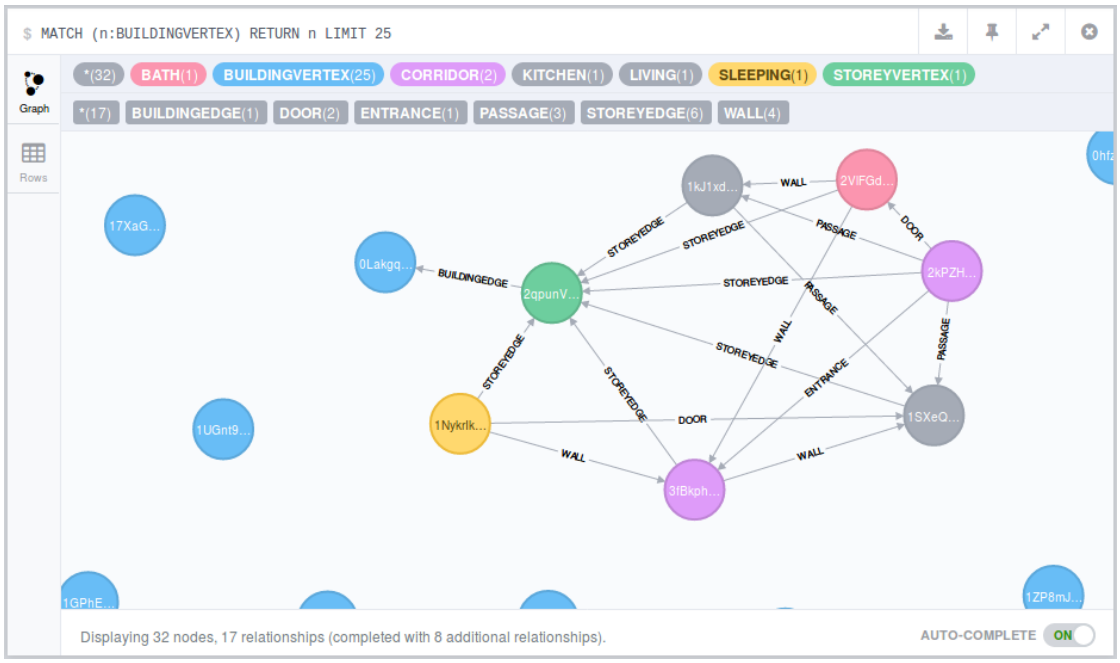


Figure 5.7.: Screenshot of the Web Interface of the Neo4j Database Showing a Typical Entry of the Project's DB.

A Neo4j database is used to organize the (see. fig. 5.7) floor plans as search results in the Metis project. These DB entries uniformly uses image URIs that are transmitted by the retrieval systems. These image URIs point to thumbnail images used in the WebUI. Each database entry corresponds to a `storeyvertex`. Each room of the database entry points with a `storeyedge` to the `storeyvertex`. The types of the room nodes themselves as well as the connections between the room nodes carry the types as defined in AGraphML.

5.2. The Archistant WebUI

5.2.1. Requirements

Apart from the methodology of digital conceptualization mentioned in section 4.1, the following requirements are regarded in the implementation of the Archistant WebUI:

Description The WebUI serves the purpose of assisting the architects to develop a building concept (apart from that, laymans should also in principle be able to use the software). This involves the need to draw and edit rooms as well as their connections. The WebUI is designed based on the room schedule working method and the idea of iterative refinement of rooms during the sketching process. An emphasis is put on the

idea that each element of a room should be expressible as abstract or specific as desired be the user. Likewise, aspects should be allowed to be reworked. Furthermore, the interface should be easy to use, fitting the human user and his/her cognitive abilities. All functions should be self-explanatory, and the software should be usable with little explanations from the outside. The software should be usable in both keyboard/mouse environments and pen-based digital devices. The WebUI should allow for polygonal room layouts. Despite that, it should not allow for free-style drawings (allowing for free-style drawings brings in a range of different additional problems that do not actually contribute to the very topic investigated in this thesis). The WebUI should be room-oriented and related to the room schedule approach.

Simplifications/Restrictions The WebUI is restricted to the generation of 2d floor plans for single-storey buildings. However, this work intends to be used for any type of building. Multiple storey buildings are approximated in the course of this thesis by using an AGraphML for each storey. As another simplification, specific wall geometries are described in polygons, restricting walls to be straight. Hence, no arc-shaped walls are allowed here. The access of all mentioned functions should be as intuitive and fast as possible. Consequently, the amount of buttons to be clicked to activate any function should be limited and the buttons should be visualized so that their function is easily visible. For that purpose, icons should be used when possible.

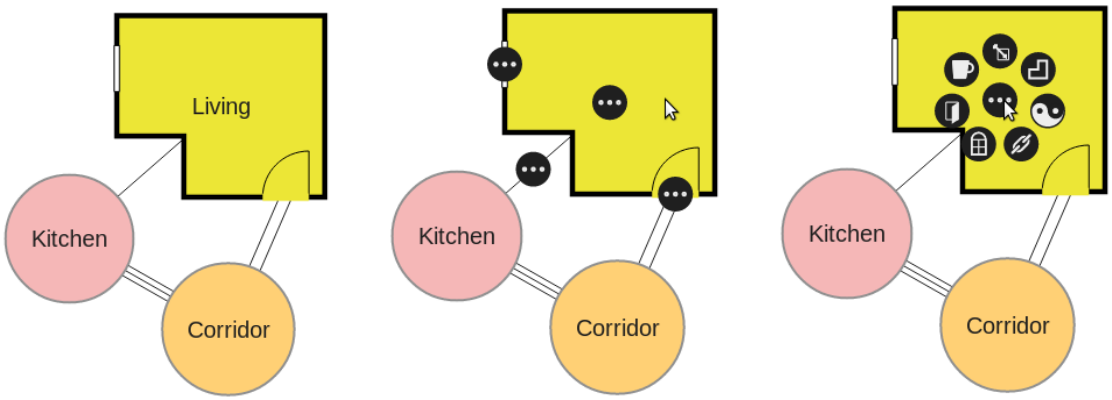
5.2.2. User Interface Elements and Concepts

This subsection describes all functions visible to the user as well as the workflow intended by the WebUI.

5.2.2.1. The Editor

The editor part of the WebUI allows for sketching floor plan concepts. These concepts may serve as search queries for reference searches. Likewise, they can stand alone. In the following, several aspects of the WebUI editor are described. The WebUI editor is *room-oriented*, hence room are the central element of the interface (many other programs are rather *space-oriented*, i.e. given space is divided into rooms by the program).

5.2.2.2. Radial Menus



(a) No Selection

(b) RM Root Buttons after Room Selection

(c) Opened RM of Room

Figure 5.8.: Interaction with the Radial Menu.

Radial menus (RM) [48] are the key interaction element of the WebUI. Whenever it is possible, the same functions are placed at the same position. For example, the delete function (indicated by a bold x) is always accessible at the right lower position of the RM. The basic interaction behavior with a room can be described as follows (see fig.5.8): If no room is selected, no RM are shown at all. If a room is focused by the user, then all RM root buttons related to that room and its RCEs are shown. This allows faster access to the functions, rather than forcing the user to first access the room itself before accessing the desired RCE. Generally, there is a trade-off in showing RM root buttons: a lesser amount of buttons force the user to make unnecessary clicks on other buttons. Too many RM root buttons may overwhelm the user and occlusion becomes more likely. After clicking on a RM root button, all other RM root buttons disappear and the selected RM opens. The user can now select a function; some buttons may carry a submenu.

5.2.2.3. Rooms

Rooms are the central element of the WebUI editor. Rooms have several properties. One of the central concepts of the WebUI is to allow to specify (and respecify) each property of a room as abstract or specific as desired.

Abstract and Specific Rooms After a room has been created, it does not have a polygon of walls yet. This state is referred to as *abstract mode* or *bubble mode*. In this mode, some of the properties may already be assigned. If walls have been specified by the room shape tool, the room is referred to as *specific* or *concrete* wall geometry mode.

In order to emphasize the difference between them, the borders of abstract rooms are thicker and more grayish than the ones of specific wall geometry rooms.

In both modes, a room can be moved seamlessly on the drawing surface individually. However, if a room has been assigned a wall polygon, the walls (edges of that polygon) snap with the walls of other rooms.

(a) Wall Shape Extension

(b) Circle of Life Extension

Figure 5.9.: Submenus of the Room's RM.

Radial Menu of Rooms Not all functions are directly available in the room's RM. The connection function is not available if there is only one room in the floor plan. Likewise, the function for drawing concrete RCEs is only available in the concrete mode. There are two submenus in the RM of a room (see fig. 5.9). These submenus try to organize functions of similar type (all functions regarding the wall polygon; all functions regarding the room's life cycle like procreation and destruction).

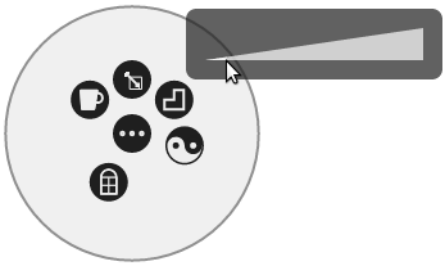


Figure 5.10.: Room Size Adjustment in WebUI.

Room Size The size of a room can be altered by an adjustment function (see fig. 5.10). This function is mainly intended to be used in bubble mode, but also available in the specific mode.

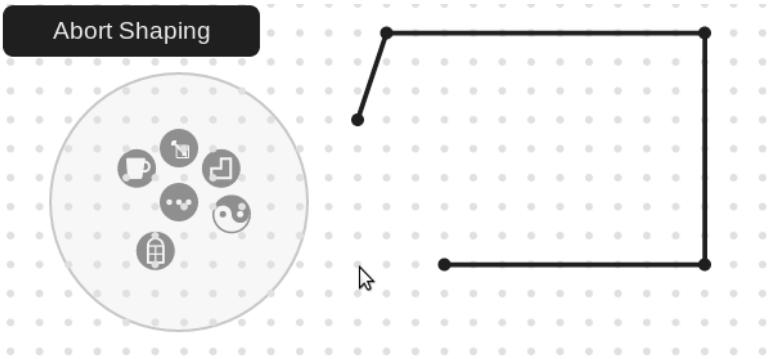


Figure 5.11.: Wall Shaping Tool of the WebUI.

Room Shape One of the most important properties of a room is the room's polygon of surrounding walls (or corners). It can be assigned by the room shape tool. With this tool, the user can draw a polygon by iteratively clicking on points of a ball grid (see fig. 5.11). The shaping tool incorporates a set of smart properties to ease the drawing process: If the user decides to make the current wall longer or smaller, the regarding point can directly be selected and the redundant point is discarded directly. Erroneous points can be erased, if the next to last point is selected. The drawing process is completed when the user clicks on the first point of the polygon for a second time.

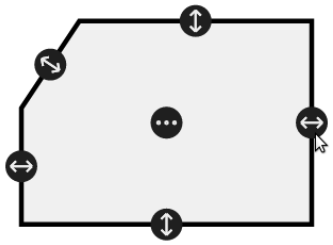


Figure 5.12.: Wall Shifting Tool of the WebUI.

Wall Shifting After a wall polygon has been defined for a room, single walls can be seamlessly shifted (see fig. 5.12)

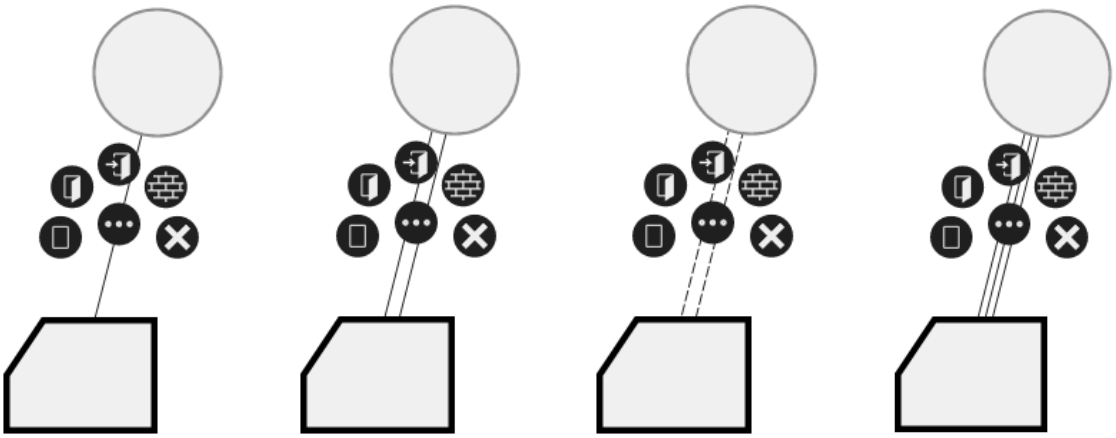
Room Function The room function (or room purpose) may be set by a group of buttons accessible in the room's main RM. The individual room function types are indicated by different colors of the room's area (e.g. yellow for living rooms). All of these colors are pastel-like.

5.2.2.4. Room Connection Elements

A room connection element (RCE) is an object which represents the part of a wall, door, a window, a passage or an entrance related to a single room. A pair of two interconnected RCEs makes up a connection between two rooms. An RCEs exists either in the RCESet of a room's point, or the abstract RCE list of a room. In the first case, the RCE is concrete, meaning that the RCE is physically determined in the room, having a certain size and position. In the second case, the RCE is referred to as abstract. An abstract RCE describes the connection between two rooms without specifying its physical dimensions. Abstract RCEs are an instance of the idea to specify aspects of a floor plan as abstract or specific as desired. An abstract RCE may serve as a reminder for the users that a connections already exist in a concept but still has to be implemented completely.

The implementation in the WebUI is a tradeoff between understandability and freedom of flexibility. A third layer of abstraction would be possible on the wall level: in such a case, the wall on which an RCE will be placed is already determined, but its actual position is not. Such an abstraction would be helpful in some cases. However, the chosen implementation already confuses some of the users.

In the current implementation, windows cannot be used for connecting rooms. They are however considered RCEs since they share many properties with other, *non-window RCEs*.

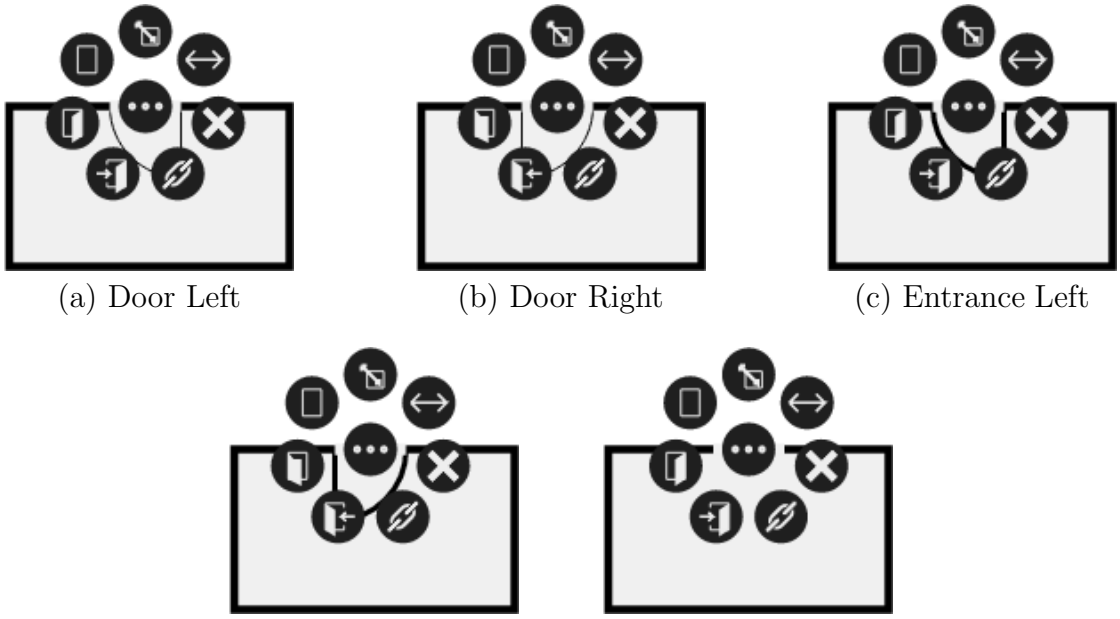


(a) Wall (b) Door (c) Entrance (d) Passage

Figure 5.13.: Link Type Visualization and their Radial Menu. This RM is only available in case of both RCEs are abstract. The different line types shown here indicate the link type and also apply in other situations.

Radial Menu of Links with two abtract RCE A link between two abstract allows for an arbitrary type definition by the user (see fig. 5.13). They can be transformed into

each other and deleted.



(a) Door Left (b) Door Right (c) Entrance Left

(d) Entrance Right (e) Passage

Figure 5.14.: RMs of Different Non-Abstract RCEs. The different kind of RCEs are transformable into each other by the RM.

Radial Menus of Doors/Passages/Entrances All different kinds of non-abstract RCEs have the same RM. They are transformable into each other by dedicated buttons (see fig. 5.14). Apart from that, a non-abstract RCEs can be altered in size, moved, deleted, and connected.

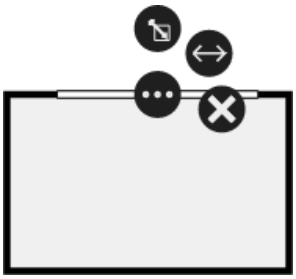


Figure 5.15.: RM of Windows.

Radial Menus of Windows Since windows are RCEs that do not connect rooms in the current implementation of the WebUI, the RM of non-abstract windows are rather simple. They only allow for altering position and size of the related window.

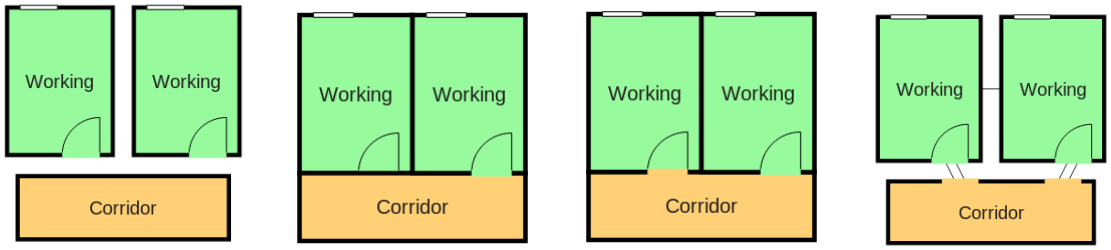
5.2.2.5. Load/Store Locally

The WebUI has been designed that after its files have been loaded to the client browser (which is completely done after the WebUI started), the most parts of the editor can be used without server interaction. Consequently, an offline usage is possible (and might be amplified in future by packing the WebUI in dedicated applications). In order to underline this usage and for the convenience of the user, AGraphML files representing the WebUI's editor content can be stored to and loaded from client's local file system.

5.2.2.6. Undo/Redo

For the convenience of the user, the WebUI incorporates a linear UNDO [39] and REDO function for most functions.

5.2.2.7. Auto Link



(a) Desnapped Rooms (b) Snapped Rooms (c) Auto-Linked Snapped Rooms (d) Desnapped Auto-Linked Rooms

Figure 5.16.: Workflow of Auto Link Function.

The room-oriented approach comes with the disadvantage that connecting elements like doors and passages need to be defined in both rooms that are linked by them. Without any compensation, this leads to a massive overhead or repetitive work in the user's workflow. This could be avoided by assuming that an non-window RCE in a wall that is snapped to the wall of a second room is automatically replicated to the other room and connection is made up between these two RCEs. But in order to allow for both a maximum of control over the sketch and to avoid unnecessary interactions, a dedicated, manually triggered auto link function (see. fig. 5.16) is used. It only takes into account

non-abstract RCEs in snapped walls that are not connected and aims for the following work flow:

1. After defining abstract rooms (and connections between them), the user successively defines wall geometries for the individual rooms. The rooms can still be moved around and remain unsnapped.

2. The user defines non-abstract RCEs (replacing the abstract ones), where one RCE for each link is defined only (i.e. if two rooms are connected via a door, only one door with adjusted dimensions needs to be added in one of the rooms).

3. The user snaps together all rooms of the sketch.

4. The user triggers the autolink function.

5. The user triggers the explosion function to check the connections

6. The user trigger undo to restore the intended room positions

5.2.2.8. Explosion

The WebUI provides an explosion function that spreads apart a floor plan by moving the centers of the rooms away from each other. The main purpose of the explosion function is to easily check the connections of a floor plan for inconsistencies. By using the explosion function and afterwards using the undo function such a review can be done with two clicks rather than manually tearing the floor plan apart and putting it together again.

In some situations, the connections can hardly be seen after an explosion. In order to mitigate this issue, the center of explosion can be altered. If no room is selected, the center of explosion is simply the center of the drawing surface. Is a room is selected, the room's center becomes the center of explosion.

5.2.2.9. Glue

In order to move groups of rooms simultaneously (and to consequently retain their relative positions), the so-called glue mode has been incorporated into the WebUI. When turned on, all rooms that are snapped to the focused room (as well as rooms that are snapped to them) are moved together with the focused room.

5.2.2.10. Grid

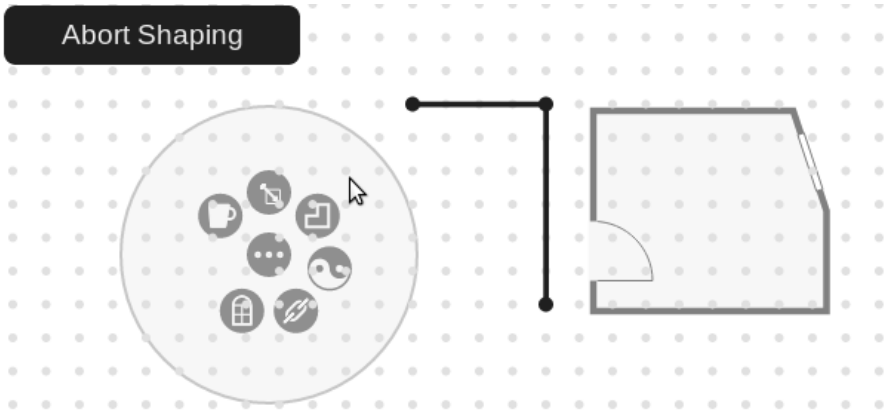


Figure 5.17.: Mismatch of Drawing Grid to an Existing Room.

Allowing for seamlessly moving rooms and the restriction that new room layouts are drawn on a fixed grid comes with the problem that new rooms may not fit to the layout of existing ones (see fig. 5.17). To point this out: room layout polygons are created with the shaping tool, which always uses the same fixed grid. After a room shape was created, the room can be moved arbitrarily. Likewise, the resize function and the wall shifting function allow for continuous altering. As a result, the corners in the drawing area does not necessarily match the grid after further user interaction. In order to mitigate that issue, a switchable grid is incorporated into the WebUI. When activated, rooms snap to the grid rather than to each other.

However, the problem could also be solved in other fashions: Besides the fixed grid, one grid for each room could be used. But this would lead to a rather chaotic, pseudo-free-style grid. Likewise, the fixed grid could be shifted and scaled so that it meets the corners of existing rooms. But multiple rooms could conflict here, and the conflict would be needed to be resolved either by the user (leading to an overhead in the user's interaction) or by an automatic strategy (that could fail to meet the user's expectations). The chosen solution is far from being perfect since the user has to move the rooms to grid manually. Nevertheless, the behavior is predictable for the user in comparison to the other mentioned approaches.

5.2.2.11. Ruler

Figure 5.18.: WebUI Ruler Visualization.

Rulers (see fig. 5.18) are a switchable function that allows for accurate measuring and analyzing the proportion of walls and RCEs. Currently, the metric system is used (i.e. lengths are expressed in meters, areas in square meters). If a wall contains no RCEs, its entire length is displayed on the ruler. If RCEs are present, there are multiple rulers per wall, one for each segment of RCE or space between RCEs and wall endings.

5.2.2.12. Creativity Function

The creativity function invokes the creativity engine described in 5.3 and is represented as a button on the left of the WebUI's editor. After the button is pushed, the current sketch is transmitted to the creativity engine, and the results are received by the WebUI. The content of the editor is then replaced with the design suggestions coming from the creativity engine. Therefore, the entire process is performed after a single click by the user and can likewise be undone by the respective function.

5.2.2.13. The Search Bar

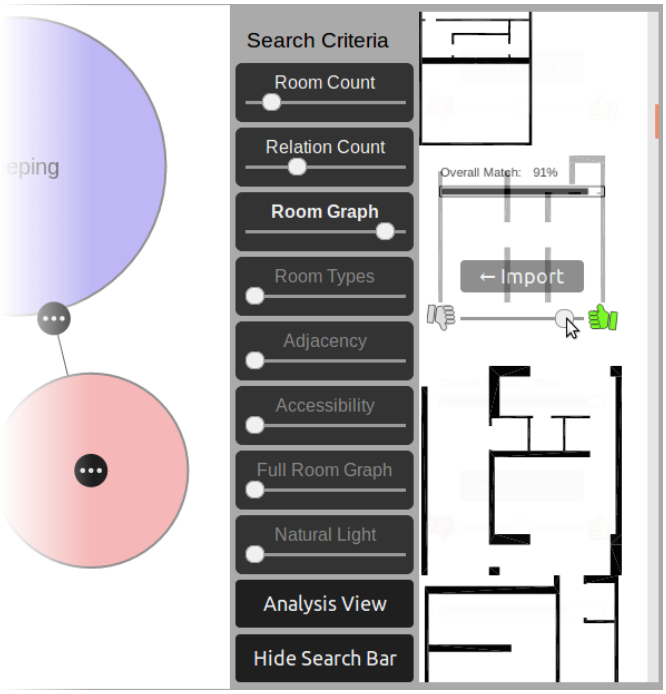


Figure 5.19.: Screenshot of the Archistant WebUI Search Bar.

The search bar allows for triggering a search, adjusting the weights of individual fingerprints as parameters of the search, contemplating the retrieval system's results and giving feedback about satisfaction for each result and to invoke the mapping view. Likewise, an import function allows to load a result's AGraphML into the editor. Finally, the function for showing the analysis view is located here.

Fingerprints that have been chosen by the user to be mandatory are indicated by a bold font (see fig. 5.19). Fingerprints that have been selected non-mandatory are indicated by a non-bold, white font. Fingerprints that have not been selected at all are presented in a grey, non-bold font.

5.2.2.14. The Mapping View

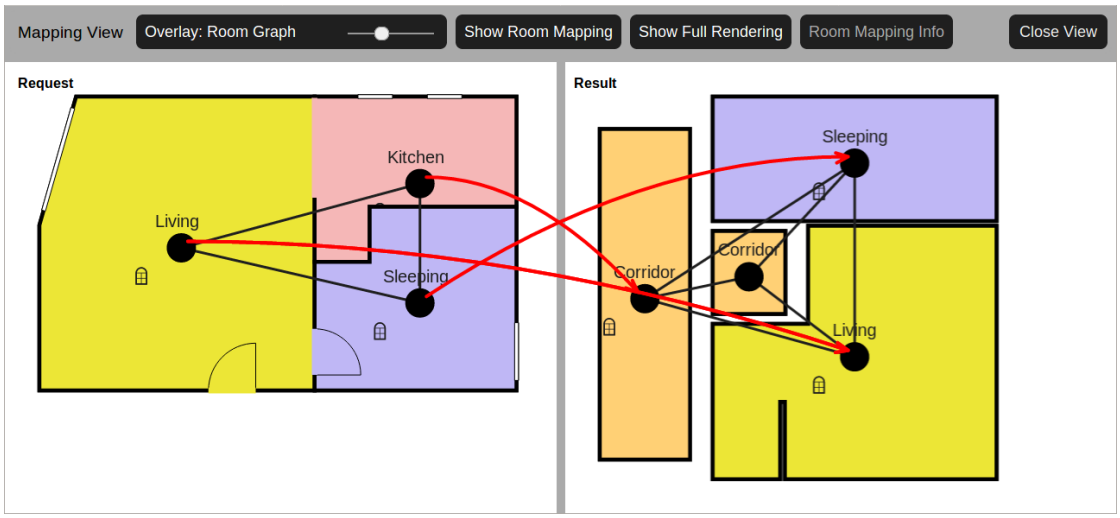


Figure 5.20.: WebUI Room Mapping Window [47].

In order to enable the user to comprehend the retrieval systems result finding process, a dedicated mapping view is integrated into the WebUI (see fig. 5.20). This full screen view allows for comparing the entered floor plan to a selected search result. Both floor plans are rendered in the background, and an overlay allows to see the semantic fingerprints of both floor plans. The mapping (displayed in red arrows) shows the mapping as generated by the AP. All of the mentioned elements can be switched on and of by dedicated buttons, the type of fingerprint can be altered by a slider. An also switchable mapping info box displays how the AS has found the mapping.

5.2.2.15. The Analysis View

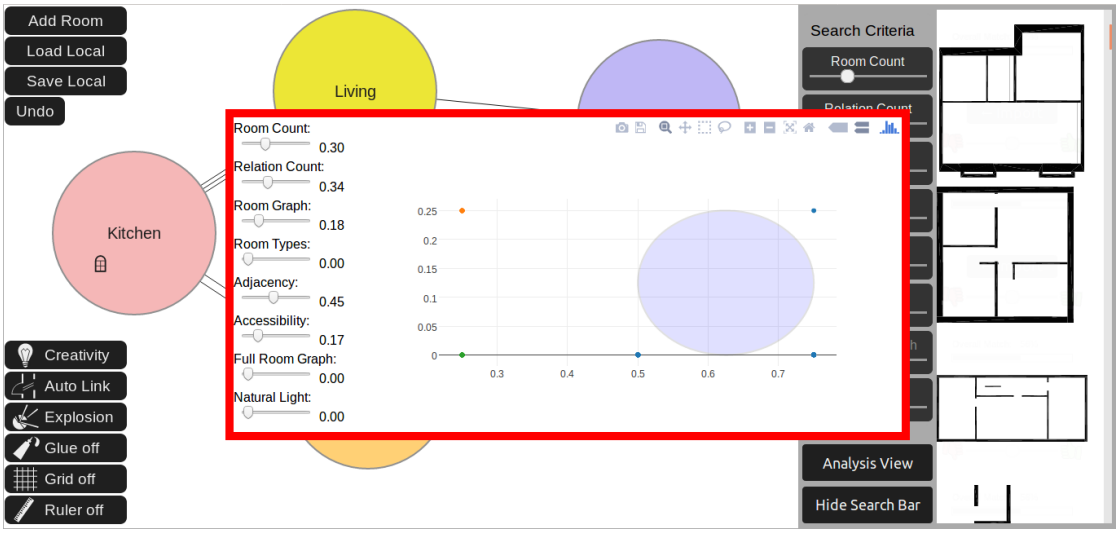


Figure 5.21.: WebUI Analysis View.

The analysis view (see fig. 5.21) shows the data coming from the analysis modules in a two-dimensional plot. It is intended to help the user finding correlations between search results and to skim through result lists more efficiently. It can be activated and deactivated by a dedicated button in the search bar. The analysis view shows the individual search results visualized as points that are arranged inside the two-dimensional plot. The formula that controls this visualization can be adjusted by sliders. The visualization makes use of a dedicated ploty [10] library.

5.2.3. Technical Implementation

In the following, some key points of the technical implementation of the WebUI editor are revealed. Several constructs of the WebUI code are referred to as *classes* for the sake of simplicity. Technically, JavaScript does not have proper classes rather than prototype-based pseudo classes.

5.2.3.1. The Room Class

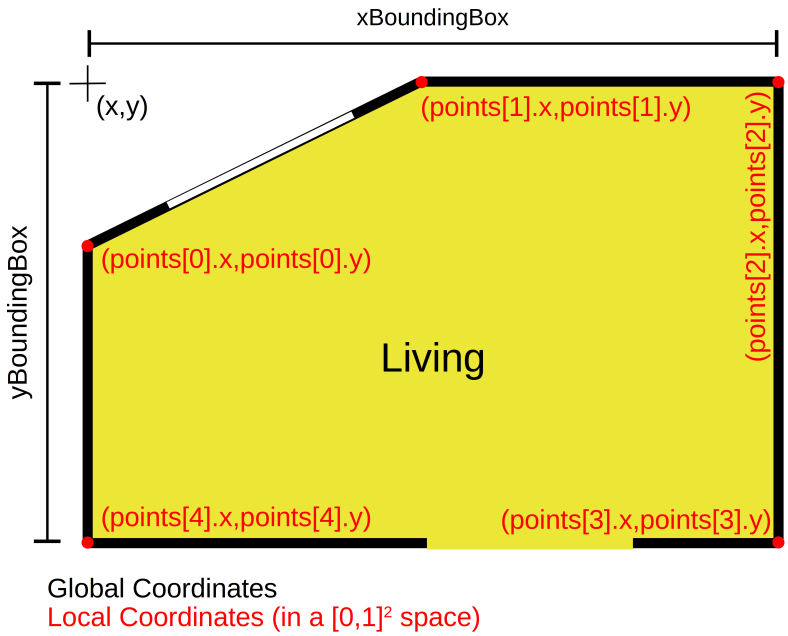


Figure 5.22.: Different attributes of the `Room` class make up an exemplary wall polygon with five points. All labeled dimensions are direct attributes of the `Room` class.

One of the most important aspects of the `Room` class is how the points of the wall polygon are represented (see fig. 5.22). These polygons are stored in an local coordinate system of the `Room` instance. This local coordinate system is transformed into the global coordinate system of the floor plan for rendering and interaction purposes. For this transformation, an instance of the `Room` class carries an anchor point that determines the translation between the room-local coordinate system and the global floor plan coordinate system. Likewise, a room's points are scaled by making use of the bounding box members of that room. The `Room` class provides a rendering function for displaying the instance on a canvas.

5.2.3.2. The Floorplan Class

The `Floorplan` class encapsulates all functionality related to the management of a floor plan. For that purpose, it owns a list of instances of the `Room` class. Furthermore, it is responsible for the management of the connections between the individual rooms. The `Floorplan` class provides a rendering function that draws the connections between the individual rooms and calls the rendering method of the aggregated `Room` instances.

5.2.3.3. The FloorplanEditor Class

The `FloorplanEditor` class is responsible for the interaction between the user and the `Floorplan` class. This class makes up the entire editor part of the WebUI. It is instantiated once, directly after the start of the WebUI. Hence, it is responsible for the display of user interaction elements like buttons, which it renders on an HTML canvas. Likewise, it triggers the rendering of the `Floorplan` class on that canvas after modifications are made (e.g. by the user). It carries the current floor plan as one member and a list of floor plans as the history of the editor.

5.3. The Creativity Engine

5.3.1. Requirements

The creativity engine (CE) should assist the user in any phase of his work. This includes the proposal of entire design steps, the completion of design steps and the augmentation of design steps.

5.3.2. Representation of Floor Plans to ANNs

A floor plan needs to be brought into a dedicated format in order to be processed by neural networks. Since recurrent neural networks are used, this format is a sequence of feature vectors that are fed one after another into the model. The format should occupy less storage, i.e. the feature vector itself and the sequence length should be as small as possible. In the following, the ANN representation is outlined. The ANN representation mimics the workflow of an architect interacting with the WebUI and is used in any scenario later.

5.3.3. The Feature Vector

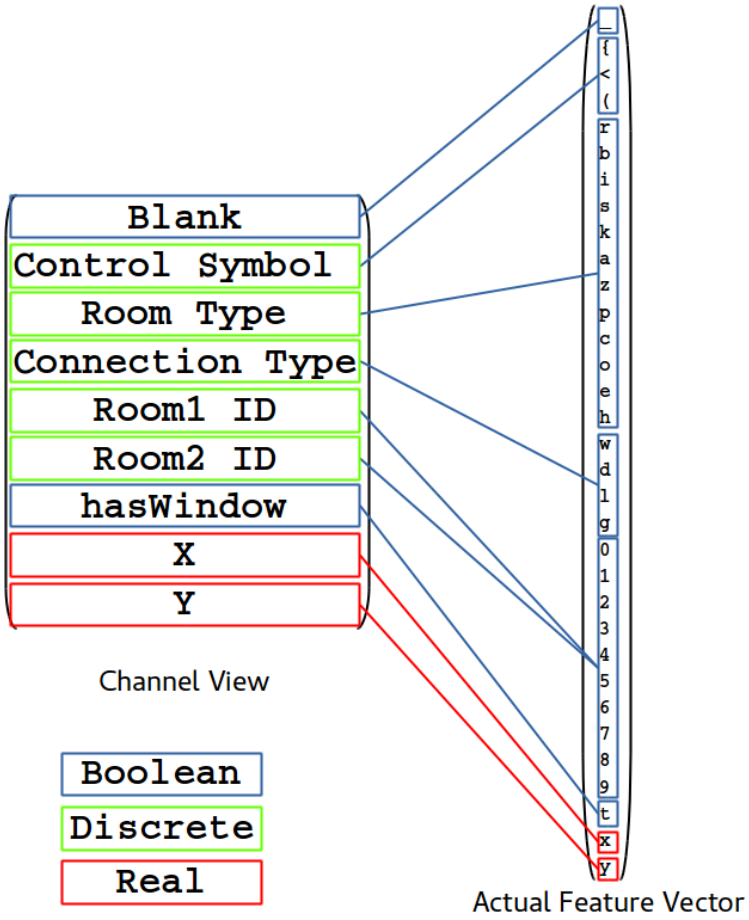


Figure 5.23.: The Feature Vector as Used in the Tight RNN Encoding. Left: The logical channels that are mapped to the vector. Right: The actual feature vector components; each vector element is represented by a different symbol.

The actual feature vector used in the tight encoding scheme comprises of a fixed number of elements. As a logical abstraction and for the sake of simplicity, the feature vector can be considered as organized into different so-called channels (see fig. 5.23). Each channel carries a piece of information and can correlate with a number of feature vector elements but every feature vector element is assigned to a single channel only.

Originally, a larger feature vector with 1-hot encoding has been considered. Numbers in that channel have been encoded by using multiple feature vectors and the decimal system and feature vector sequences were organized similar to XML (with tag opening and closing symbols for each tag). However, this 1-hot encoding scheme led to feature vector sequences matrices round 9 times larger than the ones considered here. Hence,

the chosen feature vector sequences are referred to *tight encoding*.

5.3.4. The Feature Vector Sequence

This subsection describes how a sequence of feature vectors makes up the description of a floor plan. An illustration for such a sequence for a sample floor plan can be found in fig. A.2.

5.3.4.1. Tags

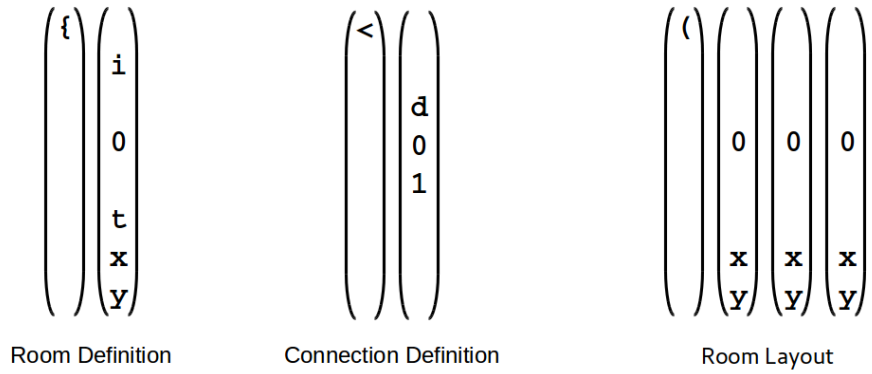


Figure 5.24.: Feature Vectors of Different Tag Types (Channel View with Omitted Blank). An empty channel space indicates that all corresponding feature vector elements are zero.

A tag is the smallest unit of a feature vector sequence. Currently, there are three different kinds of tag types (see fig. 5.24 for an overview). Every tag starts with a feature vector that carries no information but the tag types itself.

Room Defintion Tags Room definition tags define a room identified by an ID along with the room's center (expressed as a 2d-point in the euclidean space), the room's function and the `hasWindow` flag. This tag type always occupies 2 feature vectors.

Room Connection Tags A room connection tag defines a connection between two rooms by referring to their IDs and specifying the connections type. This tag type always occupies 2 feature vectors.

Room Layout Tags A room layout tag defines a room's surrounding walls by successively stating the corner's points. This tag type occupies $c + 1$ feature vectors, where c is the number of corners of the room.

5.3.4.2. Blocks

Blocks are parts of feature vector sequences which consist of tag sequences of the same kind. In the current implementation, there are 3 different blocks, that appear in the following order:

1. The Room Definition Block
2. The Room Connection Block
3. The Room Layout Block

5.3.5. Room Order

Two different approaches for the room order are proposed: A random order and a sorted order based on the room's center position. The latter can be mathematically described as:

$$a \preceq b \iff (a_x < b_x) \vee ((a_x = b_x) \wedge (a_y < b_y)) \tag{5.2}$$

This order relation orders the rooms based on their center position, primarily taking into account the x ordinate and regards the y ordinate in case of x ordinate equivalency.

The random room order tries to mimic the user's behavior in which the order in which the rooms are drawn cannot be determined a priory (advantage). The random room order is harder to compute and inherently brings problems in predicting missing tags in block 1 (disadvantage).

The sorted room order leads to a better predictability of tags in block 1, but fails to mimic the user's typical behavior. The mentioned advantages and disadvantages only affect tag prediction inside block 1. In the following, the sorted room order is used.

5.3.6. Sample Preparation

During the process of converting the floor plans from a database (where they are stored as a folder of AGraphML files) into a set of feature vector sequences, some details have to be regarded. For example, all 2D points in a floor plan (like room centers and wall polygons) have to be converted to the $[0, 1]^2$ space used in the feature vectors. Furthermore, the scarcity of manually generated database entries urges for generating multiple samples used for training out of a single database entry. To be more precisely, the converting process must be capable of generating multiple different ANN representations for a single database entry, that are fed into the ANN during learning. But also during inference, issues come up. For example, by the time the user requests a creativity suggestion, the room's individual layouts are not yet completed. Nevertheless, the room center points have to be mapped to the $[0, 1]^2$ used in the feature vectors without the knowledge of the final and total dimension of the floor plan. In all cases, the floor plan should fill the $[0, 1]^2$ space rather effectively.

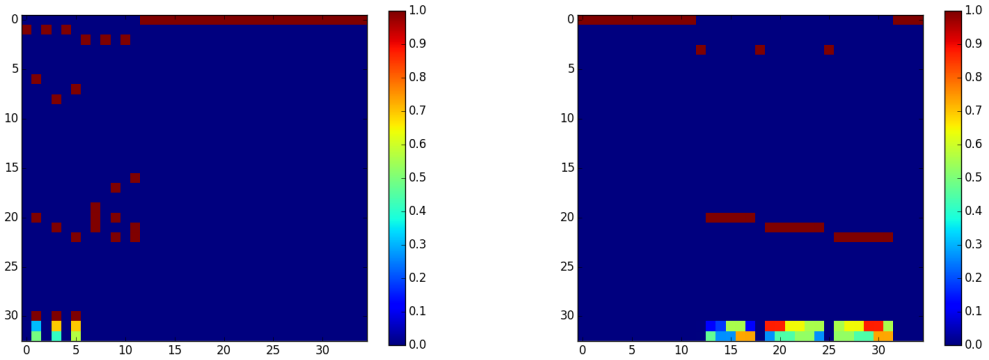
Generally, different operations are applied when converting a floor plan concept into a feature vector sequence:

- Rotation. This is done by an angle that is a multitude of 90 degrees (the resulting samples should be processable more easily and human users don't rather expect design suggestions not to be askew). A rotation by an arbitrary angle has also been considered, but in order to minimize the random sources and to meet the expectation of even room orientation, random multitudes of 90 degree angles have been chosen. As an disadvantage, the amount of samples that can be created from a single data base entry is more limited.
- Scaling. The set of points are scaled by a randomized factor that is restricted so that the larger dimension (either x or y) fills the $[0, 1]$ space by 80 to 100 percent. When no wall polygons are present, an additional margin is added in order to simulate the polygons that are added in future conceptualization.
- Translation. Within the possible space, the floor plan points are translated by a random vector.
- Noise. Room centers are noised additionally in oder to mimic the user's behavior (normally, users don't draw the rooms exactly at the final centers when doing a first, rough bubble sketch).

5.3.7. Sequencer Types

In this thesis, the process of mapping the sequence of feature vectors to a pair of input and output sequences is referred to as *sequencing*. Models trained on different sequencing methods are distinguished as different *sequencers*.

5.3.7.1. Block Generation Sequencers



(a) Input (b) Output

Figure 5.25.: Intended Behavior of Block Generation Sequencers for Block 3 on Floor Plan Sample.

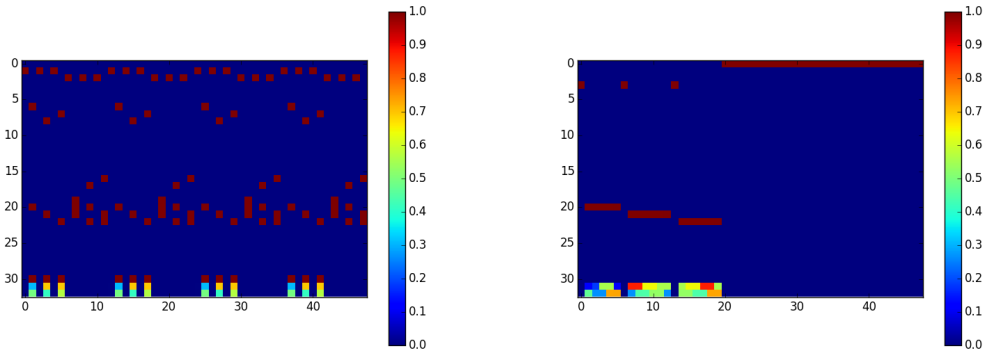
Block generation sequencers are provided the first n blocks of a floor plan concept as an input and return the $n+1$th block as output afterwards (see fig. 5.25). Block generation sequencers possess the following features:

- For each block, a model can be trained individually. Therefore, the problem is decomposed into multiple sub-problems. If a new block is added to the software system, the trained models for the existing blocks may remain usable.
- Input and output are clearly divided from each other.
- A model may take all information from the previous blocks into account while building the new block.

Nevertheless, the block generation sequencers have the followings drawbacks:

- Each model may only perform one single task.
- Multiple models must be stored.
- The model has to store a lot of information.

5.3.7.2. Block Transformation Sequencers



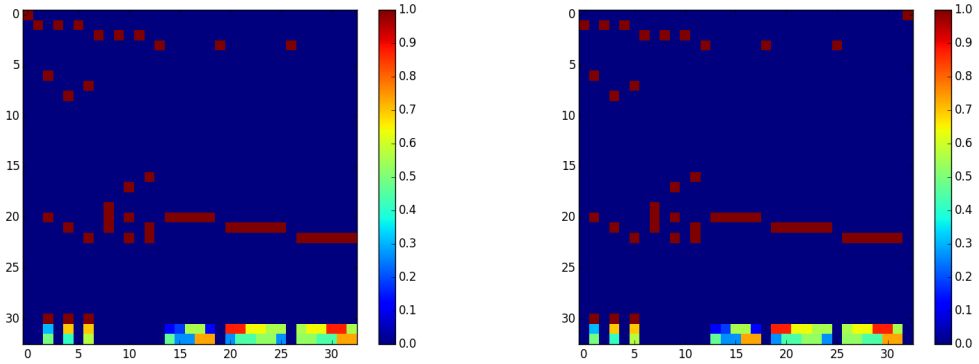
(a) Input (b) Output

Figure 5.26.: Intended Behavior of Block Transformation Sequencers for block 3 on Floor Plan Sample. The input is repeated multiple times to minimize the amount of time steps information have to be stored.

In order to overcome one of the drawbacks of the block generation, a variation of the block generation sequencers, the so-called block transformation sequencers are introduced. They are similar to the information contained in input and output, but the way they are presented differs: the information-containing part of the output it generated *while* the

input is read. More precisely, the input is repeated multiple times so that the output can fit in (see fig. 5.26). Such a setting makes the use of BIDI structures mandatory in order to make the problem computable.

5.3.7.3. Vector Prediction Sequencers



(a) Input (b) Output

Figure 5.27.: Intended Behavior of Vector Prediction Sequencers on Floor Plan Sample. The output is a one step to the past shifted version of the input. The resulting gabs are simply filled by blank vectors.

Vector prediction sequencers try to predict the nth feature vector of a feature vectors sequence given the first n feature vectors of that sequence (see fig. 5.27). Idealized they possess the following features:

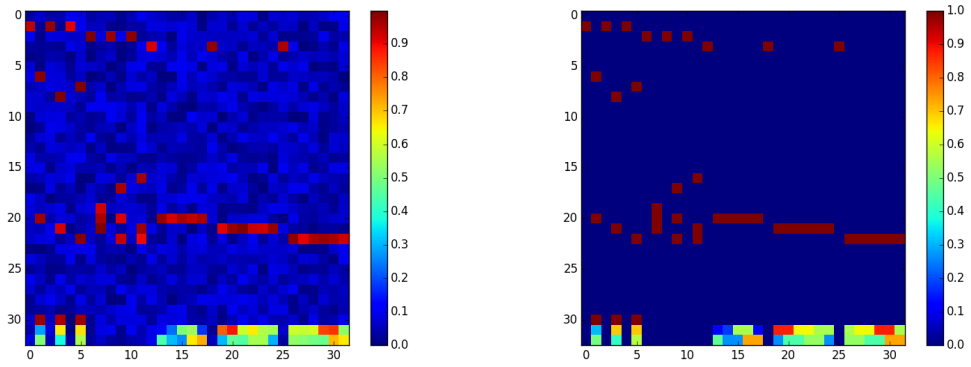
- One single model is (when trained well) capable of performing a multitude of functions (generation of different blocks).
- The model may take information from a part of a block into account, allowing for additional functions like completion of a block or prediction of a new part of a block.

Nevertheless, the vector prediction sequencers have the followings drawbacks:

- If there is a new block added to the existing software system, a new model has to be trained.
- When using in iterative structure for generating new content, the error in a prediction is again fed into the model for further predictions, resulting in an increase of errors over time.

This type of sequencer only makes sense in the context of forward networks and should not be applied to BIDI RNN structures since they could simply copy the content from the present time step towards one step in the past.

5.3.7.4. Vector Correction Sequencers



(a) Input (b) Output

Figure 5.28.: Intended Behavior of Block Transformation Sequencers on Floor Plan Sample. The input is noised and the model aims to remove the noise, thus recovering the original feature vector sequence.

Vector correction sequencers take a noised version of a feature vector sequence and try to generate one free of noise (see fig. 5.28). For that reason, the feature vectors are blended with random vectors (their maximum lengths are parameterized).

This type of sequencer can be used in two different fashions: Firstly, vector correction sequencers may be used for enhancing the result of other predictors. Secondly, the vector correction sequencers may be used to generate new parts of a feature vector sequence by appending random vectors to a incomplete vector sequence and apply the vector correction sequencer.

5.3.8. The shallowDream Structure

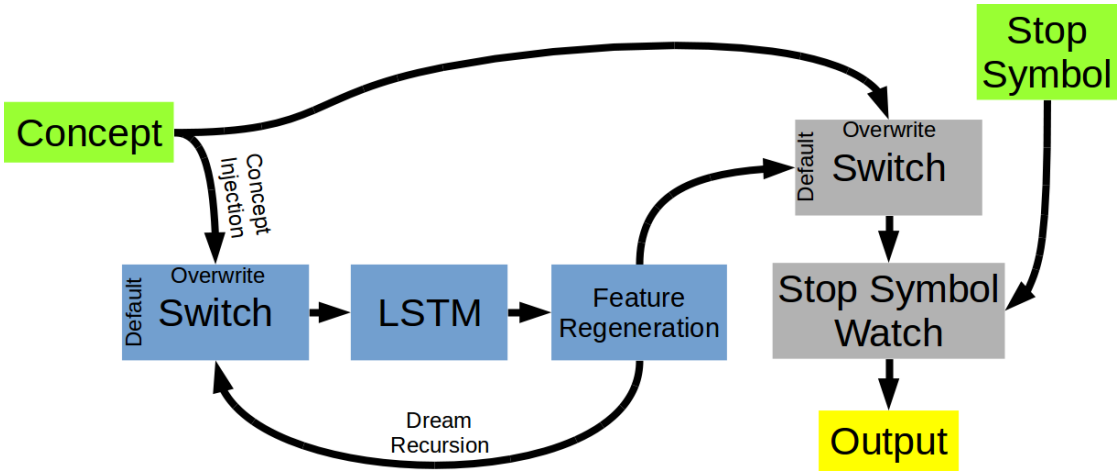


Figure 5.29.: The shallowDream Structure. This structure extends feature vector sequences from vector prediction sequencers and allows the incorporation of sequence parts from the outside.

Inferencing from block generation, block transformation and vector correction sequencers is rather trivial since the input is clearly defined and the main challenge is to cut off the blank vectors from the result. However, in order to extend feature vector sequences from vector prediction sequencer models, the most simple approach is to iteratively generate a new vector by feeding the existing sequence into the model and appending the new vector to the existing sequence. In order to be applicable, several points have to regarded:

- The input sequence has to be fed into the ANN as it is, disregarding the ANN's ouput.
- Without compensation, errors inflicted by the ANN are fed back into it over and over again. This causes an accumulation of errors over time.
- The process has to be stopped at a certain point of time.

The shallowDream structure (see fig. 5.29) tackles these issues by regenerating the feature vector, watching for dedicated and parameterized stop symbols and injecting the existing input (which is referred to as *concept*) into the ANN; simultaneously disregarding the ANN's output. The name shallowDream is chosen in referring to Alex Graves's metaphor of a dreaming person [27], with the extension that this person's dreams are influenced by inputs from the outside.

5.3.8.1. Programming the shallowDream Structure

By using different concepts and stop symbol combinations, the shallowDream structure allows for performing multiple functions using the same ANN model. A list of possible functions is given in tab. 5.1. In other words, different types of (input) concepts and stop symbols program the model for a behavior.

Function	**Concept**	**Stop Symbol**
Add a Room Given a Set of Room Definitions	n Room Definition Tags + {	{ or <
Create All Room Connections	Block 1 + <	(
Complete Room Connections	Block 1 + n Room Connection Tags + <	(
Create All Room Layouts	Block 1 + Block 2 + (	_

Table 5.1.: Overview of different functions that a vector prediction sequencer can perform given different input concepts and stop symbol combinations.

5.3.8.2. Feature Vector Regeneration

In order to regenerate the feature vectors that are coming out of the LSTM for further processing (output and feed-back), different techniques are considered. More precisely, a regeneration functionality is needed that takes one vector at a time step and generates a new vector for the given one. However, this regeneration functionality is not necessarily a map in a mathematical sense, since it may have an internal memory that takes older feature vectors in consideration.

None Regeneration The most primitive idea is to simply output the given input feature vector.

Vector-Based Regeneration The vector-based regeneration is a map in the mathematical sense. It takes a vector and tries to recover the feature vector elements. However, this technique does make use of knowledge about the feature vector structuring. More precisely, the boolean feature vector elements are recovered by mapping them to their closest allowed state (values smaller than 0.5 are mapped to 0.0 and values greater than 0.5 are mapped to 1.0).

Sequence-Based Regeneration The sequence-based regeneration is based on an internal state machine that tries to track the current state the feature vector sequence in. Based on that knowledge, a feature vector is recovered.

6. Experiments

This chapter describes the experiments conducted in the course of the thesis at hand. Each section in this chapter deals with experiments regarding an individual component of the framework. After an experimental setup has been informally introduced and subsequently described formally along with the used parameters, the results obtained are outlined afterwards. Finally, remarks are given to how the experiment's results are estimated regarding their implications to the proposed software solution (how they were used to improve it). The experiments described here directly make use of the software described in chapter 5.

6.1. WebUI - Usability Study

In order to assess the WebUIs capabilities to serve as a sketching editor (and consequently a tool for entering search requests and to view search results) is assessed by the means of a user study. In such a study, human users coming from the architectural domain are advised to work with the WebUI and to give feedback about their experiences with the prototype. Likewise, it is assessed to what degree they can fulfill their tasks using the prototype. This experiment has first been described in [16].

Formal Description of Experiment A group of 15 participants from the architectural domain (students of architecture) is ordered to work on a certain design task (developing an apartment for a certain fixed price that should be built in a big German city, see fig. C.1 for the original task description). This task should be performed four times, each time in a different manner. These different modus operandi can be described as follows:

1. Design developed in a freestyle manner with pencil on paper.
2. Design developed in a freestyle sketching working method using a dedicated software prototype from TU Munich.
3. Design developed in the room schedule working method using pencil, paper and scissors.
4. Design developed in the room schedule working method using an early version of the Archistant WebUI (referred to as "Metis WebUI")

The participants are working independently from each other in a dedicated environment and had no strict time limit. The only other person present is a supervisor who

answers questions about using the prototype. While working on the tasks, the participants are videotaped. After completing the task, the participants are asked to fill out an questionnaire (see fig. C.2, fig. C.3, fig. C.4, fig. C.5 and fig. C.6 for the original questionnaire).

The answers filled in the questionnaires serves as a primary source of processed information. The questionnaire was designed to measure the different dimensions of the ISO usability definition (see Section 2.3.1). However, since the questionnaire information only reflects the opinions of the users, this data only approximates the usability dimensions. Generally, this experiment is a comparative study in which two software prototypes were compared to their traditional working method manners; thereby checking the software prototype's user acceptance. The hypothesis of the study was: "The user is able to express his thoughts with the computer tools as good as with the traditional tools."

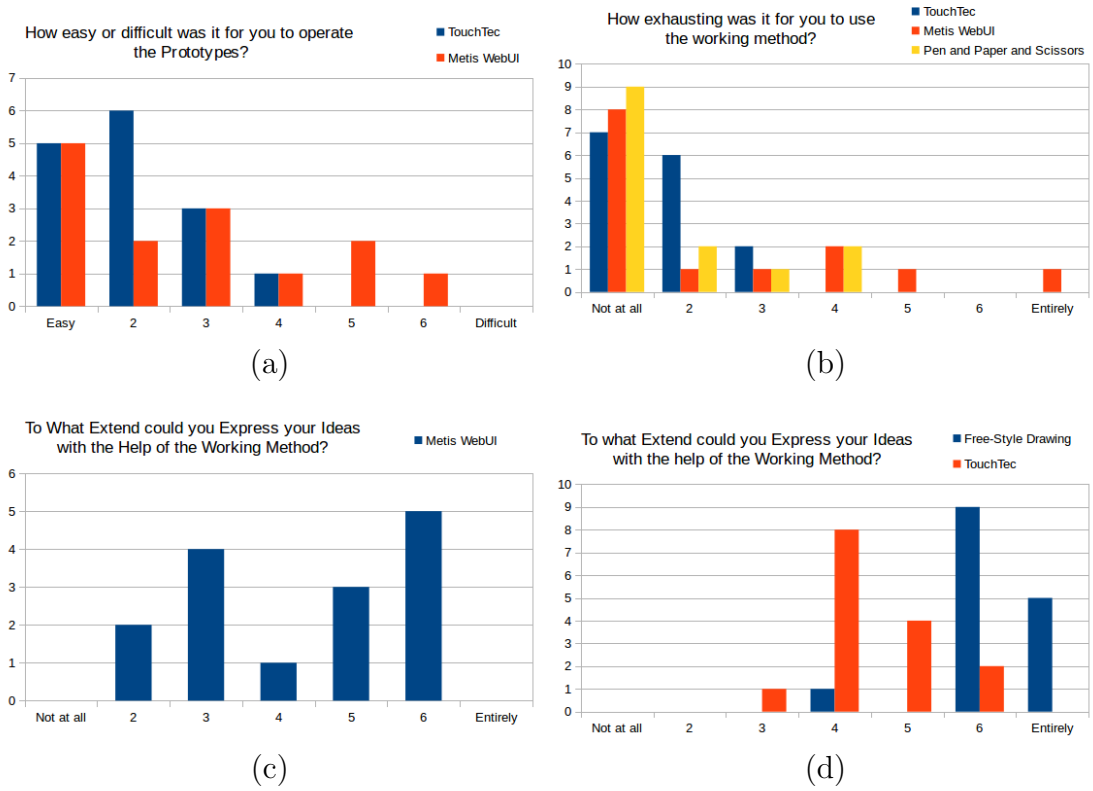


Figure 6.1.: Results of the WebUI User Study [16].

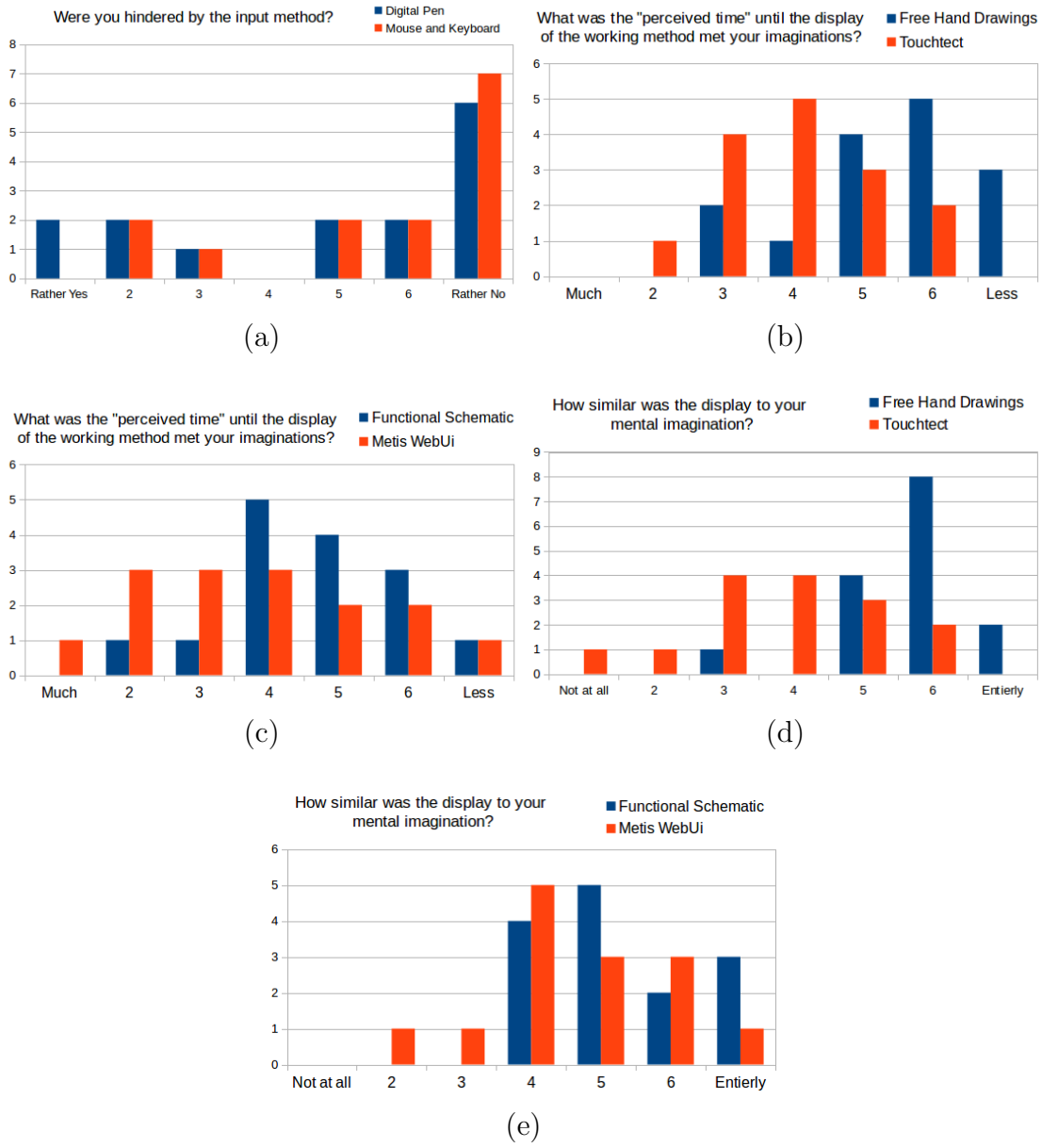


Figure 6.2.: Results of the WebUI User Study [16].

Results The accumulated responses of the participants to the closed questions are shown in fig. 6.1 and fig. 6.2. The effectiveness of the WebUI (i.e. to what degree the participants were able to fulfill their task) was measured indirectly by the questions depicted in fig. 6.1c and fig. 6.2e. The responses to these questions showed that the majority of participants were able to fulfill their tasks at least to a certain degree. The freestyle drawing prototype "Touchtect" from the TU Munich performed slightly better,

as shown in fig. 6.1d and fig. 6.2d.

The efficiency of the WebUI was vaguely assessed by asking the participants about the perceived time they needed to complete the task (see fig. 6.2c). It is measured more precisely by asking the participants how difficult they perceived the usage of the WebUI (fig. 6.1a), how exhaused they were by using the WebUI (fig. 6.1b) and how hindered they were by using the WebUI (fig. 6.2a). Generally, the results vary widely, but Touchtect performes slightly better (fig. 6.2b).

The questions for measuring the user's satisfaction in using the WebUI is also covered to a certain degree in the questions relevant for the efficiency.

Implications After finishing the user study, the supervisors reported that the participants often appeared to be slightly disappointed when switching from digital methods back to paper based ones, implying that the participants appeared open to the digital methods and that they enjoyed using the prototypes.

After this user study, the WebUI has been modified tremendously, also taking the participant's feedback into account. For example, the UNDO/REDO functions did not exist in the WebUI during the user study. Likewise, the RCE manipulation was simplified by enabling the radial menus center buttons directly when the user clicks on a room (the prototype in the study required the users to first click on a room, then open the room's radial menu, then selecting the kind of RCE to be manipulated and then to click on the RCE's radial menu button to finally do the actual manipulation).

The improved version of the WebUI (which was described in detail above) was given to three DFKI interns generate floor plan designs. They were shortly introduced to the WebUI usage and then asked to work independently. Their feedback was informally captured and is summarized as follows: All interns were able to generate reasonable floor plans by using the WebUI. The interns complained about a small bug in the autolink function. This implies that this function is of actual use in the WebUI and that the interns followed the intended workflow at least to a certain degree.

6.2. Retrieval Systems

6.2.1. Quantitative Analysis - Stress Test

A stress test is applied to the Archistant search system (PL plus attached retrieval systems). This test intends to find the limits of search query sizes (boundaries) which the individual retrieval systems can handle. This experiment has first been described in [47]. Since this test only assesses the numbers of search query sizes handleable the individual retrieval systems, this experiment is a quantitative analysis only.

Formal Description of Experiment The test case generator module of the Archistant boundary tester is used to generate a series of search queries that are processes by Archistant's processing layer and the attached retrieval systems. The actual search results generated from the retrieval system are discarded by the test case generator, but

they are recorded by the PL's logging mechanism. The log files generated during this test are analyzed by the boundary tester's analysis script afterwards. The boundaries calculated by this script are stated.

For each fingerprint a dedicated series of test search queries is used. For each such series, only the fingerprint of interest is selected as mandatory in the search requests. Different metrics are applied to assess the complexity (size) of a test search query. For most fingerprints, simply the amount of rooms in the query is used as metric. For the edge count fingerprint, the amount of edges in the test queries is used.

The boundary of a retrieval system for a fingerprint is considered the lowest complexity rating of a test query belonging to the fingerprint's query series that created an error in that retrieval system minus 1.

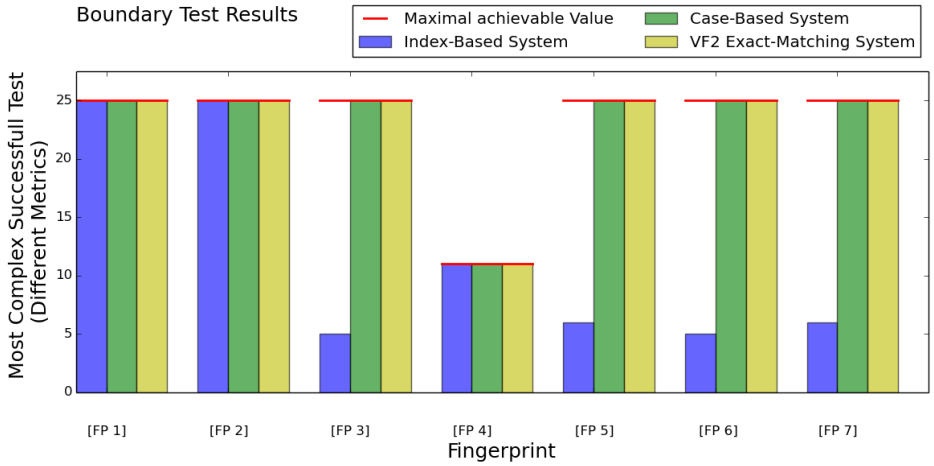


Figure 6.3.: Boundary Test Results as Calculated by the BT Analysis Skript (from [47]).

Results The results which have been calculcated by the BT analysis skript are depicted in fig. 6.3.

Implications The test generally showed that all three tested retrieval systems were at least capable of replying to some of the generated test search queries. Nevertheless, the weaknesses of the IB retrieval became obvious.

6.2.2. Qualitative Analysis - Result Adequacy Study

In order to examine how reasonable the results from the retrieval systems are (and therefore to analyze their performance qualitatively), a couple of search requests are manually created and the retrieval results are judged subjectively by a group of study participants. This experiment has been described in detail by Sabri et al. [47].

Formal Description of Experiment A series of 10 different hand-crafted floor plan concepts is successively given to the Archistant retrieval system. The results of each retrieval system for all of the search queries are shown to a group of study participants. For each query, a participant had to rank the result list according to which he liked most. These rankings of the study participants are accumulated for each search query.

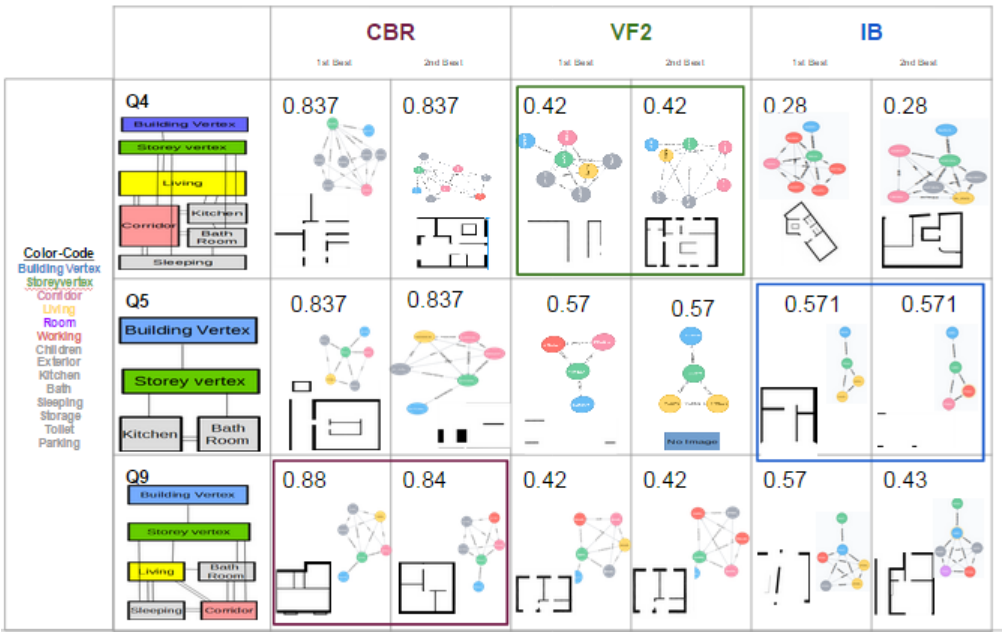


Figure 6.4.: Poll Results of the qualitative Analysis based on Subjective Estimation (from [47]).

Results For each retrieval system, there was a search query that won the majority vote by the study participants. An illustration of one of the won polls for each system including the results delivered by the systems that lost the poll in this situation is depicted in fig. 6.4.

Implications The combined retrieval system of Archistant is capable of delivering reasonable results.

6.3. Creativity Engine

6.3.1. Quantitative Analysis - Machine Learning Performance

Assessing the performance of the creativity engine automatically and by mathematical means is related to the assessment of the performance of the underlying ANN models. As a standard proceeding for scientifically applying neural networks to a problem, a

database of manually created samples is divided into a training set, a validation set and a test set (see fig. 6.5). The training itself is conducted in a series of so-called epochs, where each epoch is a different random permutation of the training set samples. While the model is trained with the samples from the training set, the error produced during forward propagation is measured. After an epoch is completed, the ANN is applied to the validation set and the produced error is also measured. However, only a forward propagation is used here; the ANN is not trained on the validation set. This training process results in a trained model. However, two models are stored for each training process: the final model that was generated by applying the full sequence and the model with the lowest error on the validation set.

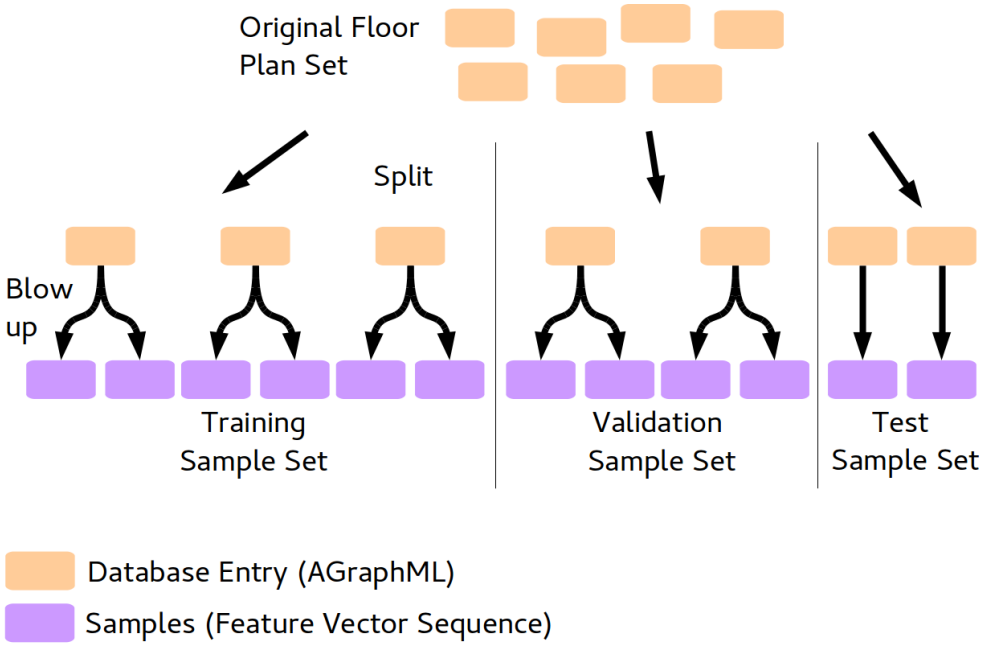


Figure 6.5.: A database of floor plan entries is divided into a training set, a validation set and a test set. Afterwards, the entries for each set are turned into samples, where each entry is turned into multiple samples, determined by the blow-up factor.

Since the training process depends on the random permutation, the random initialization of weights and ANN parameters like cell sizes and learning rate, multiple models with different parameters are trained. The model with the lowest error on the validation set is selected and consequently applied to the test set. The error resulting from this application is considered to be the final error. Furthermore, a confusion matrix calculated on the test set error can be used to assess the performance of the model more precisely and to determine the types of error that occurred.

The floor plan database (like all individual floor plans in it) used for this experiment has been dedicatedly created for this experiment. The floor plan samples from the re-

trieval database have not been used since their purpose and overall quality and room layout definitions did not meet the criteria of this experiment (in the experiments conduced here, room layouts are placed directly next to each other, while in the retrieval database, there are gaps between the rooms to enclose the intermediate physical wall).

Formal Description of Experiment A set of 75 floor plans is turned (by using a blow-up factor of 90) into a set of 6750 samples. Likewise, a set of 2 floor plans is turned (by also using blow-up factor of 90) into a set of 180 samples. Different ANNs, each consisting of a stack of layers, where four kinds of layer types are considered, are trained with the vector prediction approach. The three layer types are: LSTM, BIDILSTM, perceptron layer. For calculating the training error and the validation error, the mean of the absolute error of all feature vector component error on all feature vectors is used. The resulting learning curves and minimal errors on both sets are stated.

The model which achieved lowest error on the validation set is applied to the test set (consisting of 20 samples that are generated from 20 floor plans). As an error metric, the connection generation function described above is used: The model is used to generate the connections of a floor plan given a set of room definitions. The adjacency matrices of both ground truth and predicted connections is calculated and compared. For each connection that was not calculated correctly (equal type), an error of 0.0 is used, for each incorrectly predicted connection, an error of 1.0 is used. The sum of errors is calculated and divided by the adjacency matrix size for the final error.

Finally, a confusion matrix based on adjacency matrices differences in ground truth and predicted connections on the test set is stated. In this confusion matrix, the for each class (type of room connection) the likelihood is expressed with which an instance of this class is predicted correctly or mistakenly predicted as other classes.

When calculating the test set error and the confusion matrix for vector prediction sequencers, a shallowDream structure with sequence-based feature vector regeneration is used.

Results In the following, annealing is a factor with which the learning rate is of the ANN is multiplied after each epoch (the learning rate decreases exponentially). If two or more layers are stacked, their cell counts are divided by commas (" **,** "). Furthermore, each stack is finished with a perceptron layer that ensures an output size equal to the feature vector size. The learning curves of the vector predictor sequencer are depicted in appendix chapter D. The resulting lowest error measurements on the training set and the validation set of these training processes are listed in tab. 6.1.

Sequencer Type	Cell Count	Learn Rate	Mom.	Anneal	Nbr	Lowest Training Error (%)	LTE Epoch	Lowest Validation Error (%)	LVE Epoch
vectPred	150,150	0.01	0.3	0.8	1	0.687	47	1.908	10
vectPred	200,200	0.01	0.3	0.8	0	0.595	48	1.874	12
vectPred	300,200	0.01	0.3	0.8	1	0.553	28	2.043	4
vectPred	300,200	0.01	0.3	0.8	2	0.550	29	1.883	3
vectPred	300,200	0.01	0.3	0.8	3	0.548	29	1.893	24
vectPred	300,200	0.02	0.3	0.8	0	0.464	12	**1.789**	11
vectPred	300,200	0.03	0.3	0.6	0	0.620	12	1.932	4
vectPred	300,200	0.03	0.3	0.7	0	0.505	12	1.798	3
vectPred	400,300	0.01	0.3	0.8	1	0.494	19	1.886	4
vectPred	400,300	0.01	0.3	0.8	2	0.498	16	2.038	6
vectPred	400,300	0.01	0.3	0.8	3	0.494	17	1.869	4
vectPred	400,400	0.01	0.3	0.8	0	0.455	49	1.885	11
vectPred	400,400	0.01	0.3	0.8	1	0.454	49	1.967	3
vectPred	800	0.01	0.3	0.8	0	0.452	19	2.069	3
vectPred	800	0.01	0.3	0.8	1	0.453	21	2.211	5

Table 6.1.: Overview over the lowest error measurements on the training set and validation set achieved during training. Stated hyperparameters include learn(ing) rate, mom(entum) and anneal(ing). The lowest overall error on the validation set is indicated in boldface.

The lowest error on the validation test during training of a vector prediction sequencer was achieved by a stack of 300 and 200 LSTM cells, using a learning rate of 0.02 and a momentum of 0.3 and an annealing rate of 0.8. The achieved error was **1.789%**, measured after epoch 11. The resulting model is used for further analysis.

The error on the test set for the selected model is **62.694%**. The confusion matrix for this model calculated on the test set is depicted in fig. 6.6.

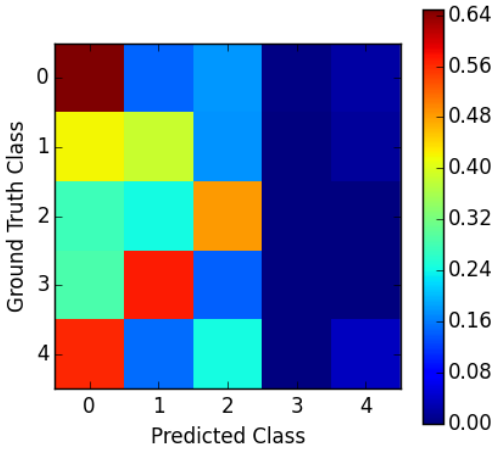


Figure 6.6.: Confusion Matrix of the finally selected vector prediction sequencer model calculated on the test set. The classes are numbered as follows: 0-No connection, 1-Wall, 2-Door, 3-Entrance, 4-Passage

Implications While the error on the training set and validation set appears low at first glance and the error on the test set appears high, both numbers have to be handled with care. First of all, they cannot directly be compared to each other due to their different calculation methods. These different methods serve different purposes. The error calculation on training and validation set intends to be a minimum computational overhead and a general assessment of the model's prediction capabilities. In contrast, the error calculation method for the test set intends to assess the actual meaning of the feature vectors produced by the model (at least the connection generation capabilities are covered).

When considering the training and validation error measurement, the following problem should be regarded: only a small amount of the feature vector accounts for the most important information (point positions). In combination with the large number of 1-hot encoded feature vector elements, the error function is potentially spoiled since these elements are usually 0 and therefore easy to predict.

The test set error measurement was particularly hard for the models: Incorrect predictions resulted as complete failure even for connection types that may be considered similar in further analysis. Hence, the fact should be considered that in contrast to most classification problems, there might not always be one correct answer to a creative and generative problem like floor plan generation.

6.3.2. Qualitative Analysis - Performance Case Study

The creativity engine's performance is illustrated by showing exemplary results generated by the creativity engine. This experiment is done using the existing Archistant

system. The creativity engine functions are generally triggered by simply hitting the creativity button in the WebUI; the creativity engine then selects the adequate function automatically. However individual rooms in the floor plan concepts processed by the creativity engine sometimes need to be moved manually for clearness reasons.

Formal Description of Experiment Using the finally selected vector prediction model from the previous experiment, the creativity engine functions are successively applied to sample data. More precisely, a set of rooms is used as origin. Initially, the connection generation function is triggered. Then, the resulting graph is used as input for the layout generation function. Screenshots of all steps are shown. After each step, the individual rooms are manually rearranged for clearness reasons (i.e. to resolve occlusions). During this experiment, the sequence-based feature vector regeneration is employed in the shallowDream structure.

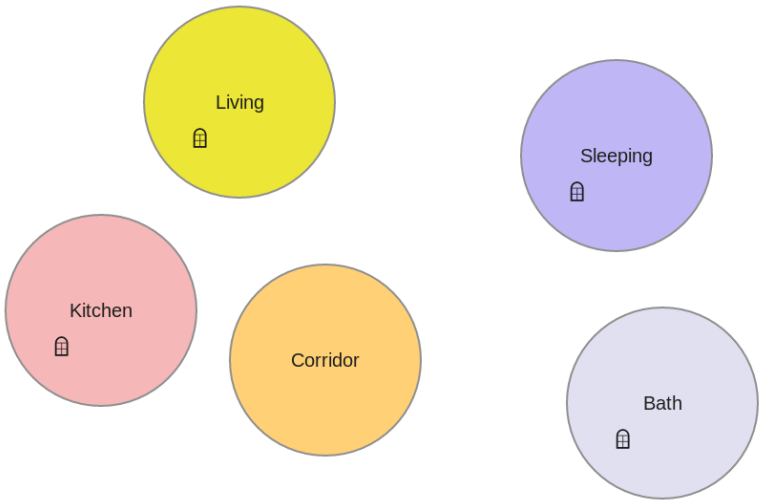


Figure 6.7.: Original Room Set

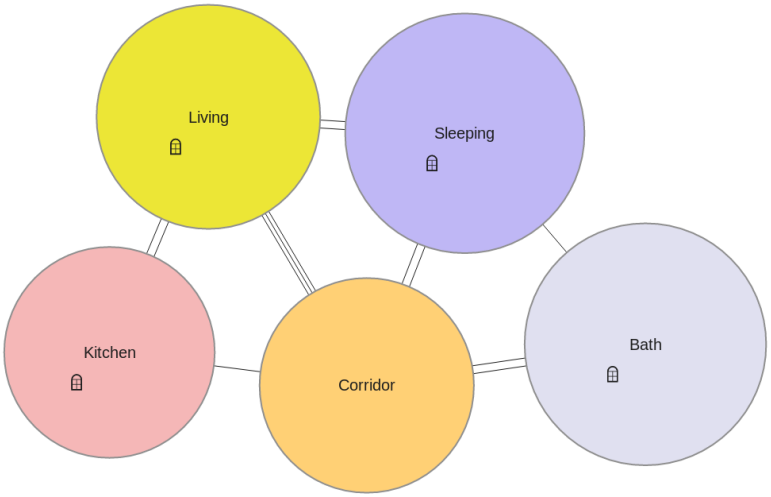


Figure 6.8.: Generated Connection Graph

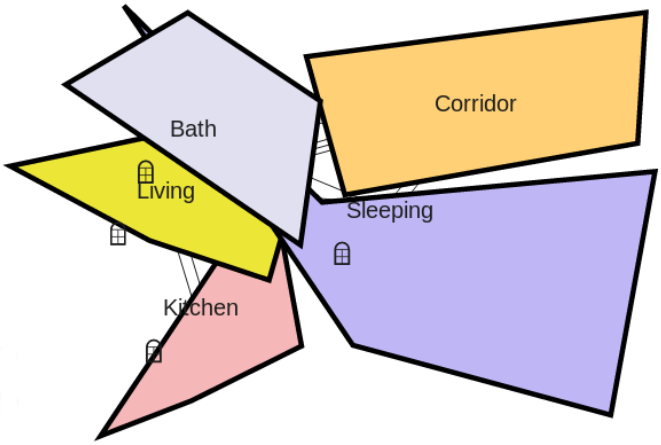


Figure 6.9.: Generated Room Layouts.

Results The screenshots are depicted in fig. 6.7, fig. 6.8 and fig. 6.9.

Implications The performance of the models has to be improved in order to be actually operational in real-world use. However, the case study illustrates the basic viability of the approach. Using the shallowDream structure, the two shown functions are both realized using the same ANN model.

7. Conclusion

This chapter describes the archived state of the developed software solution as well as discusses issues that still exist and that can be tackled in upcoming and improved versions of the software solution. The latter can be considered open research problems. For each main component described in the thesis at hand, there is a subsection describing the state of the software and there is a subsection describing future work. Apart from that, general issues of the framework are discussed in the beginning of the two main sections of this chapter.

Generally, this work dealt with the design of a software framework for supporting architects in their early design phases, avoiding monotone and repetitive labor. By utilizing different basic technologies and sciences like human computer interaction, artificial neural networks and graph theory, a modular framework has been developed that provides the user with a sketching tool, similar references designs as well as with design suggestions.

7.1. Achieved Performance

Generally, many aspects of the developed framework are fully functional. However, some functions are only working under restricted circumstances or only in a rudimentary manner, as pointed out below.

7.1.1. User Interface

The user interface of the Archistant framework, the WebUI, is fully functional. It can be used to formulate floor plan concepts, trigger search processes, view and assess their results, and to invoke the creativity engine as well as incorporating the generated design suggestions in the workflow of the UI. In the current version, full crashes are extremely rare.

However, some functions are unformed. The resize function for example inherently destroys the order established by the grid-based polygon sketching tool. More precisely, since the resize function is seamlessly, resized rooms often do not match the discrete wall lenghts of non-resized rooms. Likewise, it is next to impossible for a user to reizse two rooms exactly the same factor.

The autolink function does not yet work as expected in all situations. This might be due to the fact that a connection is only established by the autolink function if the line segment describing a wall is part of the snapped wall's describing line segment. However, overlapping walls are often also expected to be handled by this functions.

In its current form, the analysis view is more considered a rather prototypical function that has to be extended in upcoming versions.

Finally, the walls are treated as by the WebUI as if they had no physical width.

7.1.2. Retrieval Systems

The retrieval systems are functional in different degrees. In simple cases, results are returned. In one case during internal testing we retrieved a result in which all rooms but one were exactly matched to same room functions. However, the one room in which there was a mismatch between search query and result could well be reassigned a purpose from a human perspective. This subjectively hinted that how inspiration can actually be grasped from the contemplation of search results.

The IB retrieval regularly crashes in scenarios where the number of rooms in the search query exceeds a certain number. More precisely, scenarios with more than 3 rooms are critical already. These failings are caused by the connected Neo4j database and relate most likely with the number of matching possibilities that have to be considered by Neo4j. Unfortunately, such errors negatively effect the rest of the system since the augmentation processor is also connected to the same instance of Neo4j. If the hypothesis of memory problems proves to be true, the problem is simply solvable by providing more computational resources. By using multiple Neo4j instances, the negative effects of the IB to the AP can however be resolved by using dedicated instances for the individual modules.

Nevertheless, if the IB returns results, these are ensured to respect the user's wishes regarding mandatory fingerprints (the search query's fingerprint graphs are subgraphs of the result ones).

7.1.3. Creativity Engine

The creativity engine is only functioning in a rudimentary manner. Generally, the trained models already reproduce feature vector sequences that sufficiently reproduce the intended sequence syntax in order to be interpretable by the subsequent post-processing mechanics. However, the produced concepts don't resemble meaningful architectural structures. Currently, there are two pitches to improve the performance: Firstly, the used database is extremely small. By adding entries to the database, the model's performance should improve according to known principles of ANNs. Secondly, the cells sizes of the models have been limited by the the performance of the used OCRolib implementation. Using more sophisticated ANN frameworks that utilize state-of-the-art hardware accelerators can dissolve this issue.

7.2. Future Work

The current framework can only model single-story buildings. One approach to model entire buildings can be to combine multiple single storey floor plans. In order to fully support this aspect, references between multiple floor plans that indicate for example how

to precisely traverse between different storeys (by stairways, elevators and escalators) have to be incorporated.

Likewise, an appropriate *environment* for managing user accounts and collaborative work functions would be needed for a real-world deployment.

7.2.1. User Interface

The resize function of the WebUI could be improved by a smart snapping functions, that snaps the resize factor when the size of individual walls approaches the sizes of other walls. However, there is usually a bigger amount of such walls, hence the set of walls of other rooms taken into account has to be limited by a selection. Different selection techniques like selection by proximity of other rooms and taking into account the angle of a wall to the coordinate system of the floor plan have to be investigated while taking the user's responses into account in order to construct a useful function.

The same proceeding applies to the wall shifting function.

Wall thicknesses and related properties could be modeled by assigning each wall of a room a certain thickness by default, and a merging technique for wall thicknesses of two rooms that share a wall have still to be developed. This can particularly be useful when a room needs to have walls of certain properties (e.g. the walls of a bedroom might shield sound from the outside for the users comfort).

Currently, the WebUI is limited to 2D-views of floor plans. 3D-rendering could help the architects to comprehend their design decisions have on how their drafts are perceived by the habitants of their buildings.

7.2.2. Retrieval Systems

The set of fingerprints used to retrieve floor plans can be extended in order to incorporate additional floor plan aspects and hence improve the quality of the results. For example, geometric aspects are not yet covered by the existing fingerprints. Consequently, a user will not affect the result list when the sizes and amounts of walls of the rooms are altered. However, in order to match those aspects, techniques beyond graph matching are needed (for example, *point set registration* could be used).

As an methodological problem, floor plans appropriate for inspiration might not necessarily be too similar to search queries. More precisely, the user might not want to find the same floor plan he entered, since inspiration should augment the existing concept. Likewise, in rather early sketches, certain aspects like wall geometries are simply not available. Hence, there is nothing with which possible results can be compared to. All in all, the question arises whether or not only complete floor plans should be in the database or if rough sketches should be incorporated as well.

The existing database for floor plan retrieval is still small. This problem could be mitigated automatically by uploading search queries submitted by users into the database. Such a processing of course has be to brought into agreement with the users for legal reasons. Furthermore, such a modus operandi could hint how the described framework could be deployed for productive use: A crowd-based approach could be chosen.

Currently, the user has to assign the fingerprint weights manually to signal that certain aspects of his floor plan concepts matter to him. However, some users may be confused by the selection mechanism or aren't willing to manually select fingerprint weights. So in order to simplify the search process, fingerprint weights could be assigned automatically. This issue has already been considered in the course of this thesis and solutions based on the user's gaze or the time a user spends on creating as well as reworking individual aspects could be the basis for further research in this area.

The actual usefulness of search results for the user of the architectural still has to be investigated more intensively. Likewise, an approach for automatically improving search results - perhaps even tailored to the needs of individual users - is desired. Approaches to accomplish both automatical analysis and result improvements could be done with machine learning based on the already existing user's feedback function. For that purpose, the generated log files provide sufficient information.

7.2.3. Creativity Engine

The upper limit of the number of rooms in a floor plans that can be processes by the creativity engine is limited by the design of the feature vector. More precisely, for each room ID available, there is a component in the feature vector. This design aspect has been chosen since it appeared to be simple for the neural network to reference individual rooms (previously, room IDs based on the decimal system have been considered, however a protocol based on such a numbering enforces the ANN to learn the semantics of the decimal system; an overhead that should be avoided).

In order to overcome this issue, a new approach has been considered: Instead of a single neural network that generates the entire floor plan, a cluster of individual neural networks can be used, where each instance of the model took care of one of the rooms in the floor plan.

Currently, only few floor plan aspects are covered by the proposed feature vector sequences. Information about the positions and dimensions of doors/passages/entries are not yet covered. Likewise, three dimensional aspects are not yet regarded, e.g. how different storeys of the same building be described by a single feature vector sequence.

Appendices

A. Sample Floor Plan in Different Representations

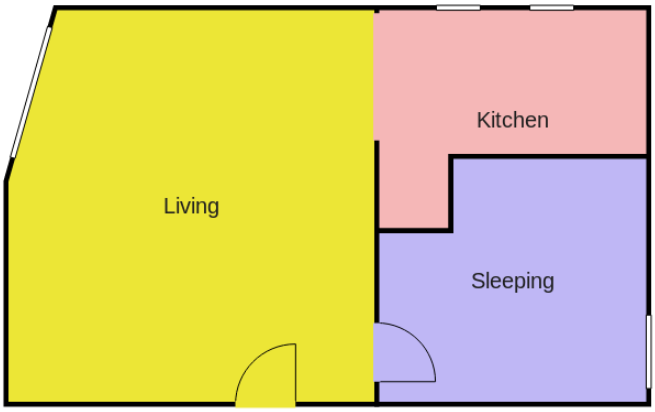


Figure A.1.: Sample Floor Plan Image (rendered in WebUI).

Listing A.1: Sample Floor Plan Encoded in AGraphML

```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
    ="http://graphml.graphdrawing.org/xmlns http://graphml.
    graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="searchGraph1" edgedefault="undirected">
    <key id="imageUri" for="graph" attr.name="imageUri" attr.type
       ="string"></key>
    <key id="imageMD5" for="graph" attr.name="imageMD5" attr.type
       ="string"></key>
    <key id="validatedManually" for="graph" attr.name="
       validatedManually" attr.type="boolean"></key>
    <key id="floorLevel" for="graph" attr.name="floorLevel" attr.
       type="float"></key>
    <key id="buildingId" for="graph" attr.name="buildingId" attr.
       type="string"></key>
    <key id="ifcUri" for="graph" attr.name="ifcUri" attr.type="
       string"></key>
    <key id="bimServerPoid" for="graph" attr.name="bimServerPoid"
        attr.type="long"></key>
```

```
<key id="alignmentNorth" for="graph" attr.name="
   alignmentNorth" attr.type="float"></key>
<key id="geoReference" for="graph" attr.name="geoReference"
   attr.type="string"></key>
<key id="name" for="node" attr.name="name" attr.type="string"
   ></key>
<key id="roomType" for="node" attr.name="roomType" attr.type=
   "string"></key>
<key id="center" for="node" attr.name="center" attr.type="
   string"></key>
<key id="corners" for="node" attr.name="corners" attr.type="
   string"></key>
<key id="windowExist" for="node" attr.name="windowExist" attr
   .type="boolean"></key>
<key id="enclosedRoom" for="node" attr.name="enclosedRoom"
   attr.type="boolean"></key>
<key id="area" for="node" attr.name="area" attr.type="float">
   </key>
<key id="sourceConnector" for="edge" attr.name="
   sourceConnector" attr.type="string"></key>
<key id="targetConnector" for="edge" attr.name="
   targetConnector" attr.type="string"></key>
<key id="hinge" for="edge" attr.name="hinge" attr.type="
   string"></key>
<key id="edgeType" for="edge" attr.name="edgeType" attr.type=
   "string"></key>

<node id="room0">
  <data key="roomType">LIVING</data>
  <data key="center">POINT (337.48 221.34000000000003)</data>
  <data key="corners">POLYGON ((168.73000000000002
     198.84000000000003, 213.73000000000002
     41.34000000000003, 506.23 41.34000000000003, 506.23
     401.34000000000003, 168.73000000000002
     401.34000000000003, 168.73000000000002
     198.84000000000003))</data>
  <data key="area">117956.25000000001</data><data key="
     windowExist">true</data>
</node>

<node id="room1"><data key="roomType">KITCHEN</data>
  <data key="center">POINT (629.98 142.59000000000003)</data>
  <data key="corners">POLYGON ((753.73 41.34000000000003,
     753.73 176.34000000000003, 573.73 176.34000000000003,
     573.73 243.84000000000003, 506.23 243.84000000000003,
     506.23 41.34000000000003, 753.73 41.34000000000003))</
     data>
  <data key="area">37968.750000000015</data>
  <data key="windowExist">true</data>
```

```
</node>

<node id="room2">
  <data key="roomType">SLEEPING</data>
  <data key="center">POINT (629.98 288.84000000000003)</data>
  <data key="corners">POLYGON ((506.23 243.84000000000003,
     573.73 243.84000000000003, 573.73 176.34000000000003,
     753.73 176.34000000000003, 753.73 401.34000000000003,
     506.23 401.34000000000003, 506.23 243.84000000000003))</
     data>
  <data key="area">51131.25</data>
  <data key="windowExist">true</data>
</node>

<edge id="edge0" source="room0" target="room0">
  <data key="edgeType">WINDOW</data>
  <data key="sourceConnector">LINESTRING (208.74291357327886
     58.79480249352408, 174.81132445290078
     177.55536441484736)</data>
</edge>

<edge id="edge1" source="room0" target="room0">
  <data key="edgeType">DOOR</data>
  <data key="sourceConnector">LINESTRING (377.61020833333333
     401.34000000000003, 432.3497916666667
     401.34000000000003)</data>
  <data key="hinge">RIGHT</data>
</edge>

<edge id="edge2" source="room1" target="room0">
  <data key="edgeType">PASSAGE</data>
  <data key="sourceConnector">LINESTRING (506.23
     46.34000000000003, 506.23 161.74277777777783)</data>
  <data key="targetConnector">LINESTRING (506.23
     161.74277777777786, 506.23 46.340000000000046)</data>
</edge>

<edge id="edge3" source="room1" target="room1">
  <data key="edgeType">WINDOW</data>
  <data key="sourceConnector">LINESTRING (600.48
     41.34000000000003, 560.48 41.34000000000003)</data>
</edge>

<edge id="edge4" source="room1" target="room1">
  <data key="edgeType">WINDOW</data>
  <data key="sourceConnector">LINESTRING (685.53
     41.34000000000003, 645.53 41.34000000000003)</data>
</edge>
```

```
    <edge id="edge5" source="room2" target="room2">
      <data key="edgeType">WINDOW</data>
      <data key="sourceConnector">LINESTRING (753.73
          387.03784529320995, 753.73 320.2393769290126)</data>
    </edge>

    <edge id="edge6" source="room2" target="room0">
      <data key="edgeType">DOOR</data>
      <data key="sourceConnector">LINESTRING (506.23
          327.40875000000005, 506.23 380.77125)</data>
      <data key="hinge">RIGHT</data>
      <data key="targetConnector">LINESTRING (506.23 380.77125,
          506.23 327.40875000000005)</data>
    </edge>

    <edge id="edge7" source="room2" target="room1">
      <data key="edgeType">WALL</data>
    </edge>

  </graph>
</graphml>
```

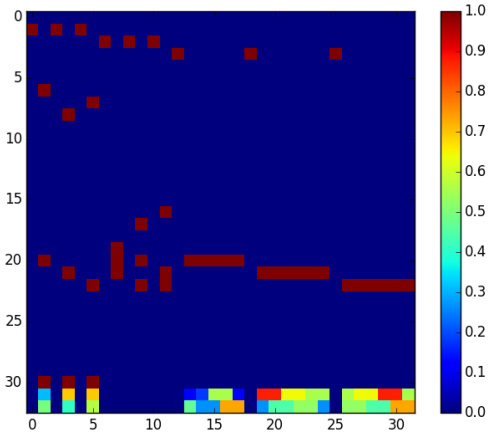


Figure A.2.: Sample Floor Plan in Tight Encoding for RNN Processing Purposes.

B. LOGXML Sample File

Listing B.1: Log XML File with two Search Requests as well as Responses from Three Servers and a Feedback to the Second Result from the User after the second search query (contained AGraphMLs as well as some of the results files omitted)

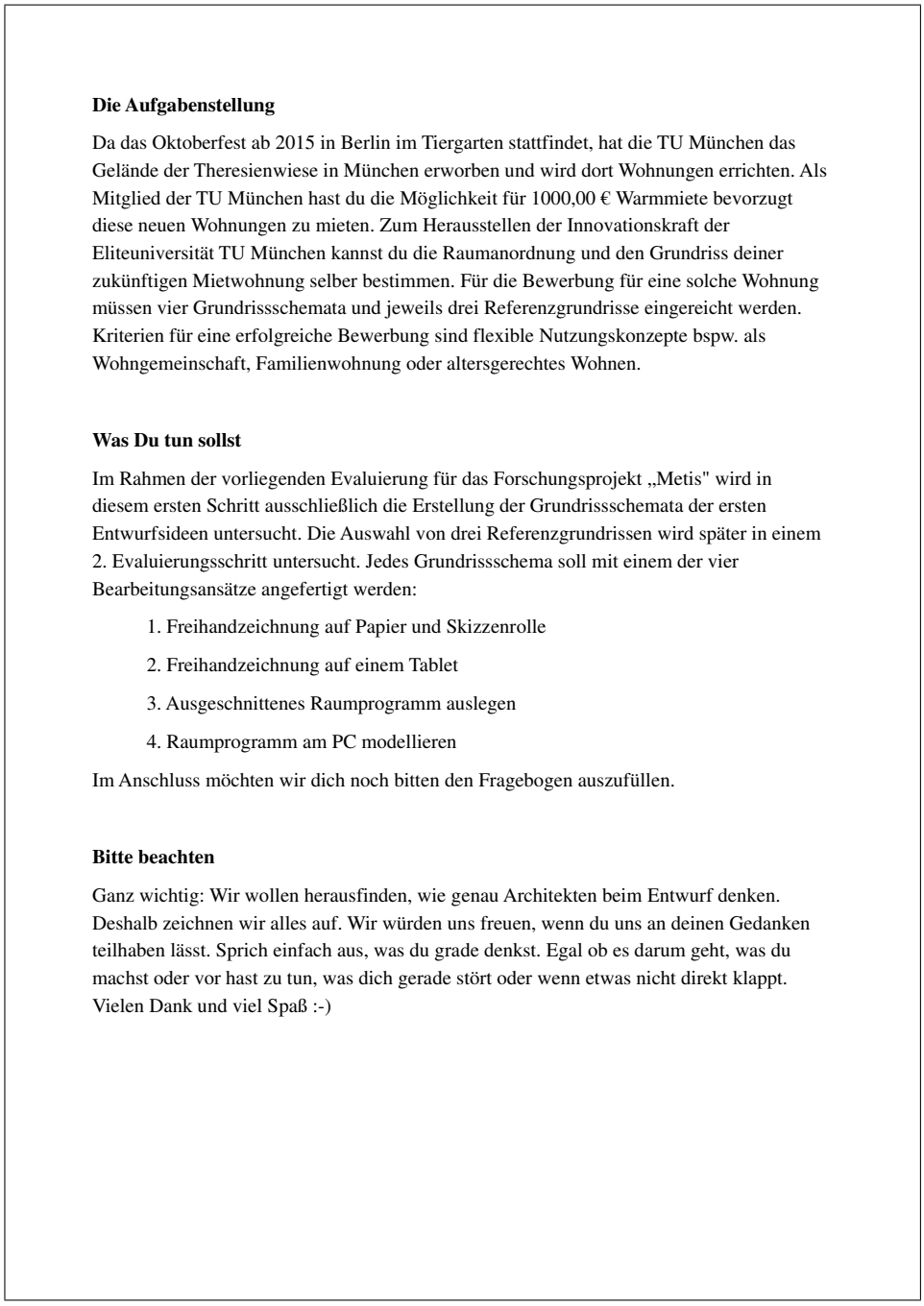
```
<log client_ip="127.0.0.1" end_time="2017-07-05T23:51:26"
    start_time="2017-07-05T23:48:19">
  <SearchOperation>
    <UserRequest time="2017-07-05T23:48:46">
      <searchRequest>
        <agraphml>
          <!-- description of search request 1 -->
        </agraphml>
        <fingerprint name="Relation_Count" weight="0.66"/>
        <fingerprint name="Room_Graph" weight="0.37"/>
        <fingerprint name="Room_Types" weight="0.68"/>
        <fingerprint name="Adjacency" weight="0.34"/>
        <fingerprint name="Accessibility" weight="0.6"/>
        <fingerprint name="Full_Room_Graph" weight="0.33"/>
        <fingerprint name="Natural_Light" weight="0.81"/>
      </searchRequest>
    </UserRequest>
    <ServerResponse systemName="Index-Based System" time="
       2017-07-05T23:49:15">
      <searchResults>
        <error>No results found.</error>
      </searchResults>
    </ServerResponse>
    <ServerResponse systemName="Case-Based System" time="
       2017-07-05T23:49:00">
      <searchResults>
        <tr>
          <td/>
          <td>0.7324061</td>
          <td>
          <matchingInfo>
            <matchedfingerprint name="Natural_Light" score="
```

```
            0.6315789"/>
          <matchedfingerprint name="Accessibility" score="
            0.6315789"/>
          <matchedfingerprint name="Relation_Count" score="
            0.9997997"/>
          <matchedfingerprint name="Room_Types" score="0.6666667
            "/>
        </matchingInfo>
        </td>
        <td/>
        <td>http://mediatum.ub.tum.de/file/1231438/126063.svg</
          td>
        <td/>
      </tr>
    </searchResults>
  </ServerResponse>
  <ServerResponse systemName="VF2 Exact-Matching System" time="
     2017-07-05T23:48:51">
    <searchResults>
      <error>No results found.</error>
    </searchResults>
  </ServerResponse>
  <FinalResultList time="2017-07-05T23:49:17">
    <resultList>
      <tr>
        <td/>
        <td>0.7324061</td>
        <td>
        <matchingInfo>
        <matchedfingerprint name="Natural_Light" score="
          0.6315789"/>
        <matchedfingerprint name="Accessibility" score="
          0.6315789"/>
        <matchedfingerprint name="Relation_Count" score="
          0.9997997"/>
        <matchedfingerprint name="Room_Types" score="0.6666667"/
          >
        </matchingInfo>
        </td>
        <td>
          <roomMap>
            <roomMapEntry from="room1" to="e29c404e-a4b0-4287-
              bc4b-35e6939c3ae5"/>
            <roomMapEntry from="room2" to="06019303-c498-4c67
              -9664-f1401ebe23f8"/>
            <roomMapEntry from="room0" to="52e9d50b-298c-4bc7-
              b29a-d570c1d1bc97"/>
            <roomMapInfo>
              8 Trying Natural_Light to create room map...
```

```
                    IMPOSSIBLE<br/>
                  4 Trying Room_Types to create room map...
                    Relation_Count cannot be used to create a room
                    map.IMPOSSIBLE<br/>
                  6 Trying Accessibility to create room map...
                    IMPOSSIBLE<br/>
                  3 Trying Room_Graph to create room map...  SUCCESS
                </roomMapInfo>
              </roomMap>
            </td>
            <td>
              http://mediatum.ub.tum.de/file/1231438/126063.svg
            </td>
            <td>
              <!-- description of search result 1 -->
            </td>
          </tr>
        </resultList>
      </FinalResultList>
    </SearchOperation>
    <SearchOperation>
      <UserRequest time="2017-07-05T23:50:58">
        <!-- like above -->
      </UserRequest>
      <ServerResponse systemName="Index-Based System" time="
          2017-07-05T23:50:58">
        <!-- like above -->
      </ServerResponse>
      <ServerResponse systemName="Case-Based System" time="
          2017-07-05T23:51:07">
        <!-- like above -->
      </ServerResponse>
      <ServerResponse systemName="VF2 Exact-Matching System" time="
          2017-07-05T23:51:00">
        <!-- like above -->
      </ServerResponse>
      <FinalResultList time="2017-07-05T23:51:17">
        <!-- like above -->
      </FinalResultList>
      <userRating rating="95" resultNo="1" time="2017-07-05T23:51:24
          "/>
    </SearchOperation>
  </log>
```

C. WebUI User Study

C.1. Task Desprition for Participants



Die Aufgabenstellung

Da das Oktoberfest ab 2015 in Berlin im Tiergarten stattfindet, hat die TU München das Gelände der Theresienwiese in München erworben und wird dort Wohnungen errichten. Als Mitglied der TU München hast du die Möglichkeit für 1000,00 € Warmmiete bevorzugt diese neuen Wohnungen zu mieten. Zum Herausstellen der Innovationskraft der Eliteuniversität TU München kannst du die Raumanordnung und den Grundriss deiner zukünftigen Mietwohnung selber bestimmen. Für die Bewerbung für eine solche Wohnung müssen vier Grundrissschemata und jeweils drei Referenzgrundrisse eingereicht werden. Kriterien für eine erfolgreiche Bewerbung sind flexible Nutzungskonzepte bspw. als Wohngemeinschaft, Familienwohnung oder altersgerechtes Wohnen.

Was Du tun sollst

Im Rahmen der vorliegenden Evaluierung für das Forschungsprojekt „Metis" wird in diesem ersten Schritt ausschließlich die Erstellung der Grundrissschemata der ersten Entwurfsideen untersucht. Die Auswahl von drei Referenzgrundrissen wird später in einem 2. Evaluierungsschritt untersucht. Jedes Grundrissschema soll mit einem der vier Bearbeitungsansätze angefertigt werden:

1. Freihandzeichnung auf Papier und Skizzenrolle
2. Freihandzeichnung auf einem Tablet
3. Ausgeschnittenes Raumprogramm auslegen
4. Raumprogramm am PC modellieren

Im Anschluss möchten wir dich noch bitten den Fragebogen auszufüllen.

Bitte beachten

Ganz wichtig: Wir wollen herausfinden, wie genau Architekten beim Entwurf denken. Deshalb zeichnen wir alles auf. Wir würden uns freuen, wenn du uns an deinen Gedanken teilhaben lässt. Sprich einfach aus, was du grade denkst. Egal ob es darum geht, was du machst oder vor hast zu tun, was dich gerade stört oder wenn etwas nicht direkt klappt. Vielen Dank und viel Spaß :-)

Figure C.1.: This figure despicts the original task description handed to the participants of the WebUI user study. This task desciption has been developed by Christoph Langenhan.

C.2. Questionaire for Participants

Übertragung von analoger in digitale Arbeitsweise - Skizze

Benutzen Sie in frühen Entwurfsphasen Skizzen, um Ihre Entwurfsideen zu formulieren?	○**Ja** ○ **Nein**

Inwieweit konnten Sie Ihre Ideen/Konzepte ausdrücken?	
Mit Freihandzeichung	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
Mit dem Prototyp "Touchtec"	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

War das Arbeiten mit dem Prototyp "Touchtec" für Sie im Gegensatz zum "klassischen" Freihandskizzieren (Papier / Stift) anders?	○**Ja** ○ **Nein**

War der Arbeitsablauf mit dem Prototyp "Touchtec" für Sie im Gegensatz zum "klassischen" Zeichnen mit Stift und Papier anders?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

Wie war die „gefühlte" Zeit bis die Darstellung ihren Vorstellungen entspricht?	
Freihandskizze	**Viel** ○ ○ ○ ○ ○ ○ ○ ○ **Wenig**
Touchtec	**Viel** ○ ○ ○ ○ ○ ○ ○ ○ **Wenig**

Wie bewerten Sie die Flexibilität bei der Anpassung der Darstellung?	
Freihandskizze	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
Touchtec	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

Wie ähnlich ist die Darstellung mit der mentalen Vorstellung Ihres Konzeptes?	
Freihandskizze	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
Touchtec	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

War etwas beim Arbeiten mit dem Prototypen „Touchtec" im Vergleich zur Arbeit mit Papier und Stift schlechter?	○**Ja** ○ **Nein**
Wenn ja, was? (max. 5 Auszählungen)	

Figure C.2.: Page 1 of the Original Questionaire for WebUI User Study Participants.

War etwas beim Arbeiten mit dem Prototypen „Touchtec“ im Vergleich zur Arbeit mit Papier und Stift besser?	○**Ja** ○ **Nein**
Wenn ja, was? (max. 5 Auszählungen)	

Benutzeroberfläche Touchtec

Wie verständlich war die optische Gestaltung (z.B. Bedeutung v. Symbolen, Schaltflächen, Funktionen) für Sie?	**Verständlich** ○ ○ ○ ○ ○ ○ ○ **Unverständlich**
Wie einfach oder schwierig war für Sie die Benutzung von Touchtec?	**Einfach** ○ ○ ○ ○ ○ ○ ○ **Schwierig**
Wie bewerten Sie das Aussehen von Touchtec?	**Unansehnlich** ○ ○ ○ ○ ○ ○ ○ **Ansehnlich**
Wie körperlich anstrengend empfanden Sie die Arbeit mit Touchtec?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

War für Sie die Arbeit mit der digitalen Stifteingabe hinderlich?	**Eher Ja** ○ ○ ○ ○ ○ ○ ○ **Eher Nein**

Akzeptanz

Inwieweit könnten Sie sich vorstellen, diese Arbeitsweise in frühen Planungsphasen z.B. zum Zwecke der Konzeptionierung einzusetzen?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

Welche Aspekte der Benutzeroberfläche genau empfanden Sie als verbesserungswürdig? Wie könnte eine solche Verbesserung aussehen?	

Figure C.3.: Page 2 of the Original Questionaire for WebUI User Study Participants.

Übertragung von analoger in digitale Arbeitsweise - Funktionsschemata

Benutzen Sie in früher Entwurfsphase Funktionsschemata, um Entwurfsideen zu formulieren?	○**Ja** ○ **Nein**
Inwieweit konnten Sie Ihre Ideen/Konzepte mit dem Prototyp "Metis WebUI" ausdrücken?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
War das Arbeiten mit dem Prototyp WebUI für Sie im Gegensatz zum "klassischen" Arbeiten mit Funktionsschemata (Papier / Stift) anders?	○**Ja** ○ **Nein**
War der Arbeitsablauf mit dem Prototyp WebUI für Sie im Gegensatz zum "klassischen" Arbeiten (Papier / Stift) anders?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
Gefühlte Zeit bis die Darstellung Ihren Vorstellungen entspricht. Funktionsschemata WebUI	 **Viel** ○ ○ ○ ○ ○ ○ ○ **Wenig** **Viel** ○ ○ ○ ○ ○ ○ ○ **Wenig**
Flexibilität bei der Anpassung der Darstellung. Funktionsschemata WebUI	 **Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen** **Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
Wie ähnlich ist die Darstellung zu der mentalen Vorstellung Ihres Konzeptes? Funktionsschemata WebUI	 **Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen** **Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
War etwas beim Arbeiten mit WebUI im Vergleich zu der Arbeit mit Funktionsschemata schlechter?	○**Ja** ○ **Nein**
Wenn ja, was?	

Figure C.4.: Page 3 of the Original Questionaire for WebUI User Study Participants.

War etwas beim Arbeiten mit WebUI im Vergleich zu der Arbeit mit Funktionsschemata besser?	○**Ja** ○ **Nein**
Wenn ja, was? (max. 5 Auszählungen)	

Benutzeroberfläche WebUI

Wie verständlich war die optische Gestaltung (z.B. Bedeutung v. Symbolen, Schaltflächen, Funktionen) von WebUI für Sie?	**Verständlich** ○ ○ ○ ○ ○ ○ ○ **Unverständlich**
Wie einfach oder schwierig war es für Sie, WebUI zu bedienen?	**Einfach** ○ ○ ○ ○ ○ ○ ○ **Schwierig**
Wie bewerten Sie das Aussehen von WebUI?	**Unansehnlich** ○ ○ ○ ○ ○ ○ ○ **Ansehnlich**
Wie körperlich anstrengend empfanden Sie die Arbeit mit WebUI?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

War für Sie die Arbeit mit Maus und Tastatur hinderlich?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

Akzeptanz

Inwieweit könnten Sie sich vorstellen, die Benutzeroberfläche in frühen Planungsphasen z.B. zum Zwecke der Konzeptionierung einzusetzen?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**

Welche Aspekte der Arbeitsweise genau empfanden Sie als verbesserungswürdig? Wie könnte eine solche Verbesserung aussehen?	

Figure C.5.: Page 4 of the Original Questionaire for WebUI User Study Participants.

Fragen zur Person	
Wie alt sind Sie?	
Was studieren Sie (Fachrichtung, Abschluss)?	
Im wievielten Semester befinden Sie sich?	
Wie körperlich anstrengend empfinden Sie die Arbeit mit klassischen Arbeitsweisen wie Papier, Stift und Schere?	**Gar nicht** ○ ○ ○ ○ ○ ○ ○ **Vollkommen**
Welche anderen Probleme sehen Sie in der Arbeit mit klassischen Arbeitsweisen wie Papier, Stift und Schere?	
Abgesehen von dem Experiment: Haben Sie bereits Erfahrung mit computergestützten Arbeitsweisen in der Architektur?	○**Ja** ○ **Nein**

Figure C.6.: Page 5 of the Original Questionaire for WebUI User Study Participants.

D. Learning Curves of Creativity Engine ANNs

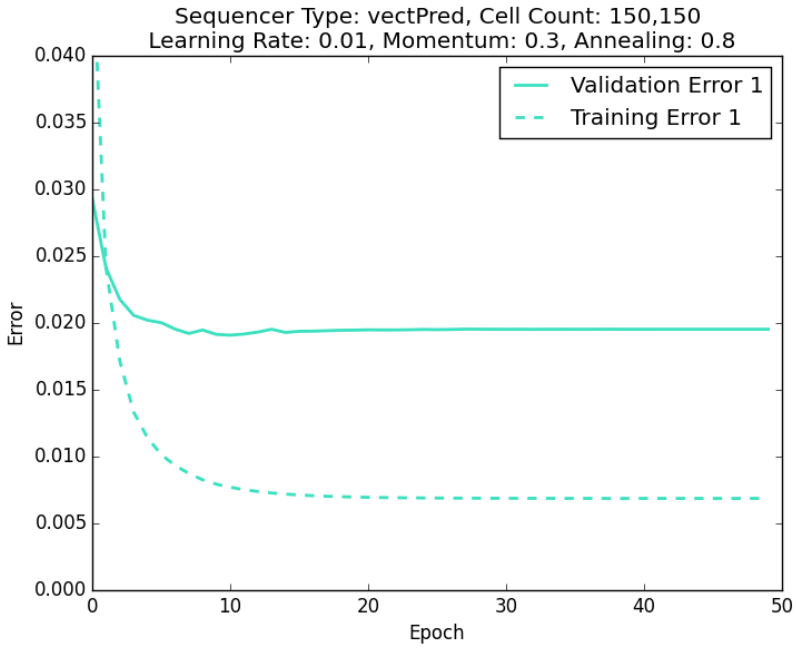


Figure D.1.: Error on training set and validation set.

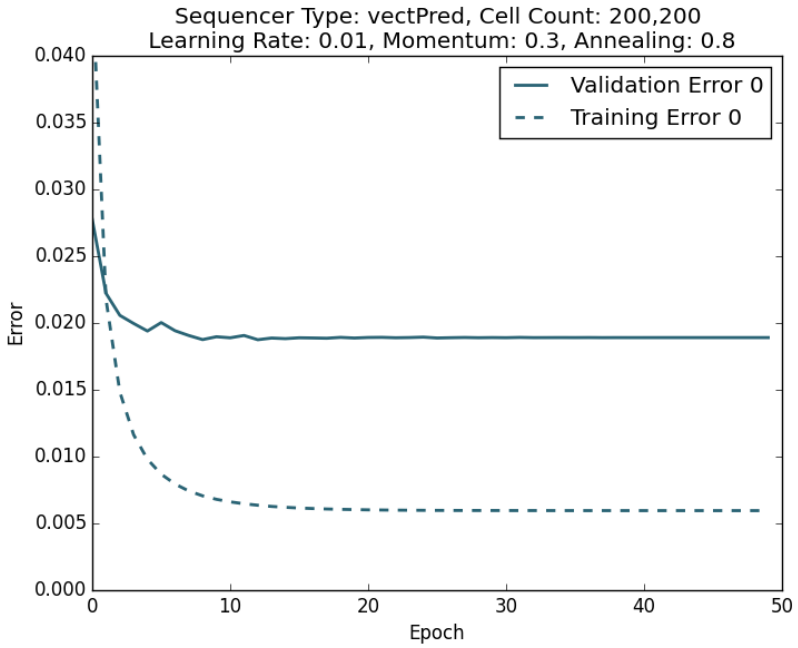


Figure D.2.: Error on training set and validation set.

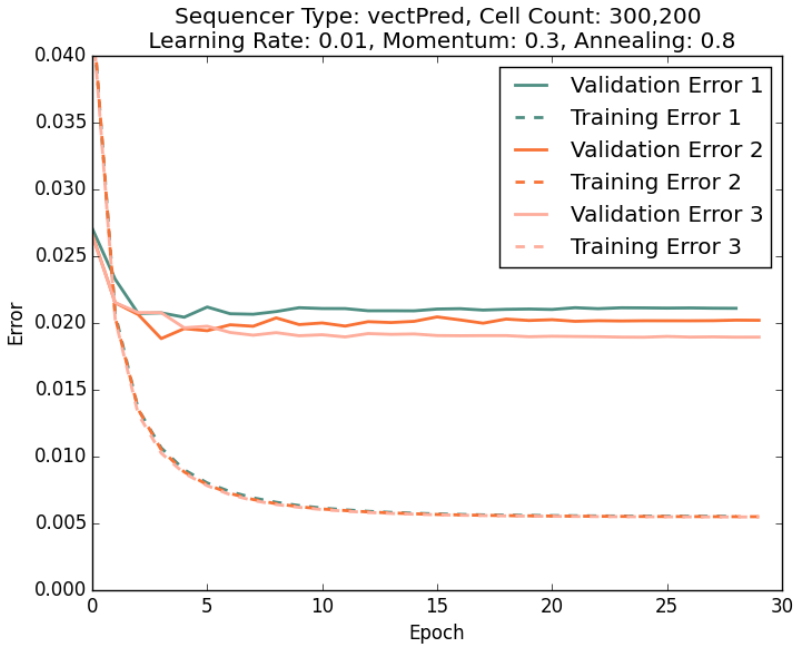


Figure D.3.: Error on training set and validation set.

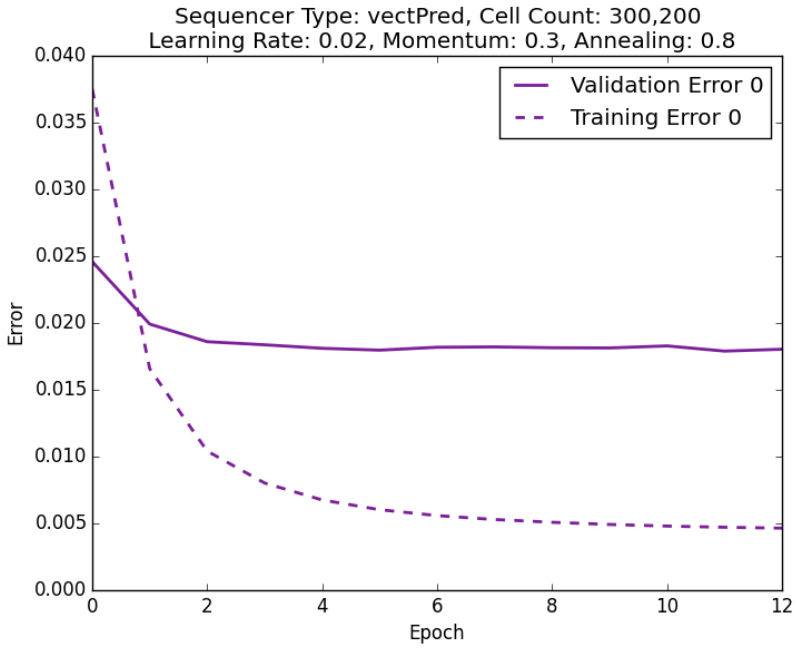


Figure D.4.: Error on training set and validation set.

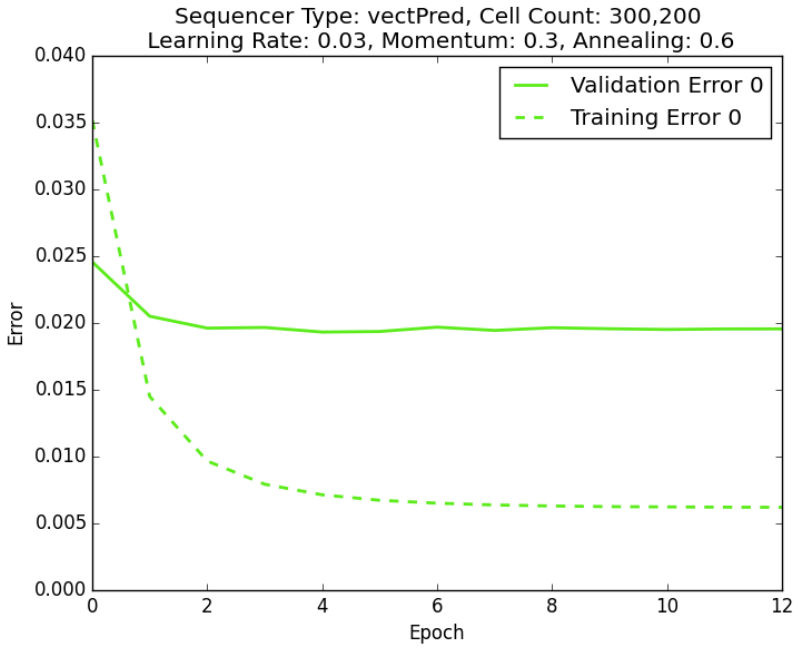


Figure D.5.: Error on training set and validation set.

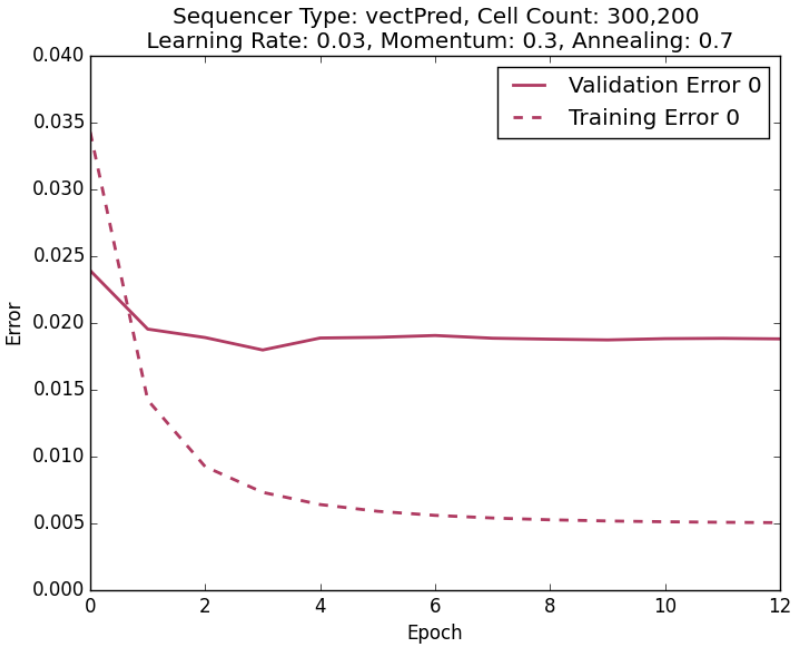


Figure D.6.: Error on training set and validation set.

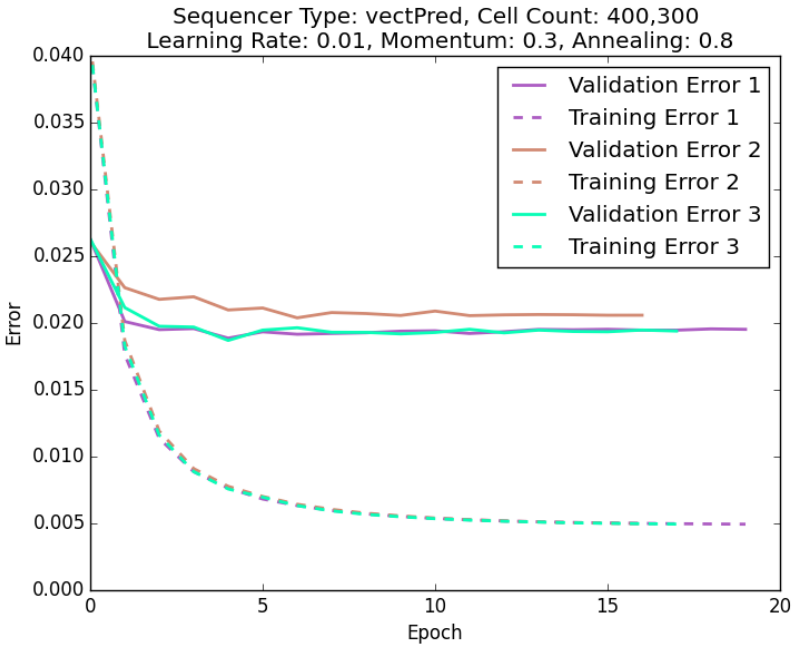


Figure D.7.: Error on training set and validation set.

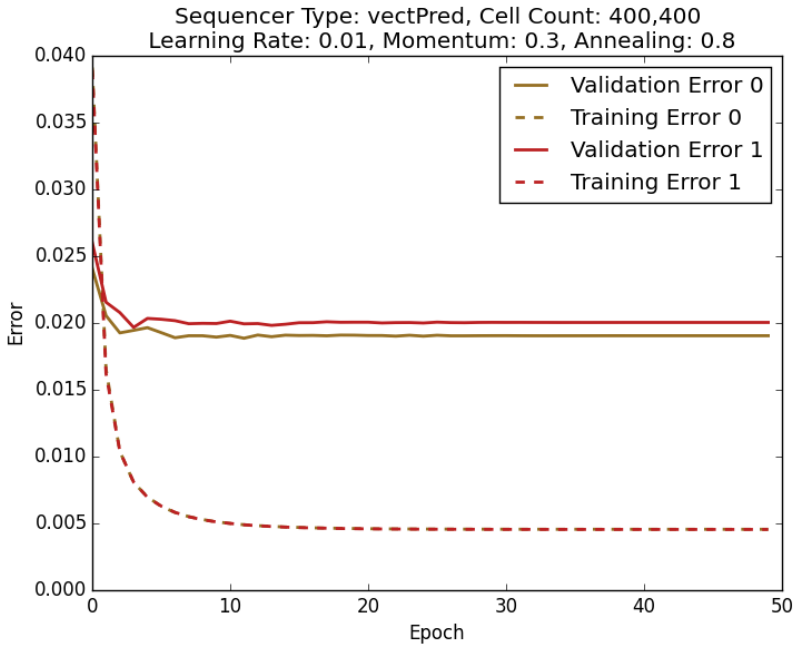


Figure D.8.: Error on training set and validation set.

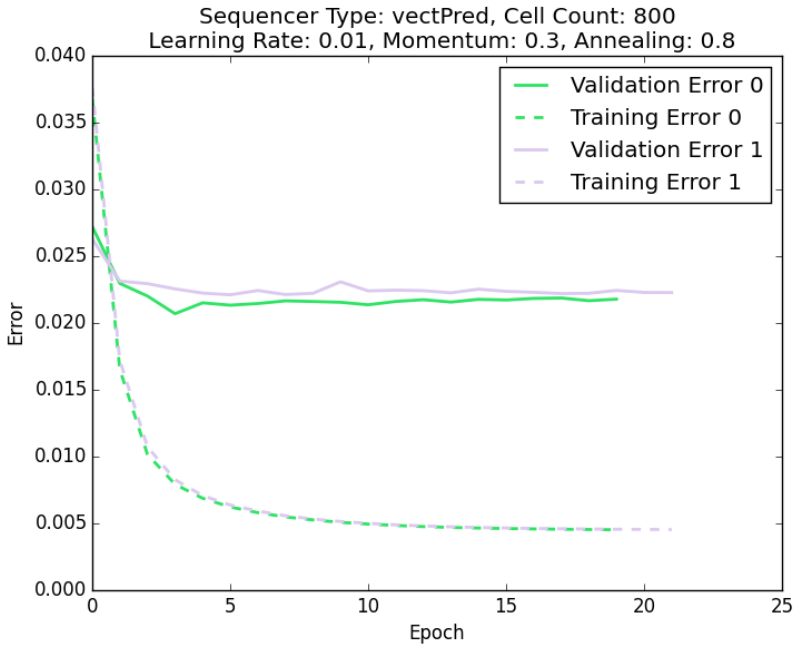


Figure D.9.: Error on training set and validation set.

Bibliography

[1] Deutsches Forschungszentrum für Künstliche Intelligenz gGmbH. https://www.dfki.de/web. Accessed: 2017-07-04.

[2] DFKI Archistant. http://www.dfki.uni-kl.de/archistant. Accessed: 2017-06-21.

[3] Firefox: Fast, private browser for desktop, iOS and Android. https://www.mozilla.org/en-US/firefox/. Accessed: 2017-05-22.

[4] International Organization for Standardization. https://www.iso.org/. Accessed: 2017-07-11.

[5] ISO 19162:2015 - Geographic information – Well-known text representation of coordinate reference systems. https://www.iso.org/standard/63094.html. Accessed: 2017-07-13.

[6] ISO/IEC 13249-3:2016 - Information technology – Database languages – SQL multimedia and application packages – Part 3: Spatial. https://www.iso.org/standard/60343.html. Accessed: 2017-07-13.

[7] Ksd research group. http://ksd.ai.ar.tum.de/. Accessed: 2017-07-04.

[8] metis Wissensbasierte Such- und Abfragemethoden für die Erschließung von Informationen in semantischen Modellen (BIM) für die Recherche in frühen Entwurfsphasen. https://www.ar.tum.de/en/research-development/projects/metis-wissensbasierte-such-und-abfragemethoden-fuer-die-erschliessung-von-informationen-in-semantischen-modellen-bim-fuer-die-recherche-in-fruehen-entwurfsphasen/. Accessed: 2017-07-04.

[9] Neuron - from wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Neuron&oldid=787323343. Accessed: 2017-07-16.

[10] Plotly. https://plot.ly/. Accessed: 2017-07-17.

[11] Filipe Afonso and Kirill Jedenov. Coding randomness: accepting unpredictability in modular systems through the use of computation.

[12] Christopher Alexander. *Notes on the Synthesis of Form*, volume 5. Harvard University Press, 1964.

[13] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1, 2008.

[14] Viktor Ayzenshtadt, Christoph Langenhan, Syed Saqib Bukhari, Klaus-Dieter Althoff, Frank Petzold, and Andreas Dengel. Distributed domain model for the case-based retrieval of architectural building designs. In Miltos Petridis, Thomas Roth-Berghofer, and Nirmalie Wiratunga, editors, *Proceedings of the 20th UK Workshop on Case-Based Reasoning. UK Workshop on Case-Based Reasoning (UKCBR-2015), located at SGAI International Conference on Artificial Intelligence, December 15-17, Cambridge, United Kingdom*. School of Computing, Engineering and Mathematics, University of Brighton, UK, 2015.

[15] Viktor Ayzenshtadt, Christoph Langenhan, Syed Saqib Bukhari, Klaus-Dieter Althoff, Frank Petzold, and Andreas Dengel. Thinking with containers: A multi-agent retrieval approach for the case-based semantic search of architectural designs. In Joaquim Filipe and Jaap van den Herik, editors, *Proceedings of the 8th International Conference on Agents and Artificial Intelligence. International Conference on Agents and Artificial Intelligence (ICAART-2016), February 24-26, Rome, Italy*. SCITEPRESS, 2016.

[16] Johannes Bayer, Syed Saqib Bukhari, Christoph Langenhan, Marcus Liwicki, Klaus-Dieter Althoff, Frank Petzold, and Andreas Dengel. Migrating the classical pen-and-paper based conceptual sketching of architecture plans towards computer tools-prototype design and evaluation. In *International Workshop on Graphics Recognition*, pages 47–59. Springer, 2015.

[17] Suresh K Bhavnani and Bonnie E John. Exploring the unrealized potential of computer-aided drafting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 332–339. ACM, 1996.

[18] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. Graph markup language (graphml). *Handbook of graph drawing and visualization*, 20007:517–541, 2013.

[19] Thomas M Breuel. The ocropus open source ocr system. In *Electronic Imaging 2008*, pages 68150F–68150F. International Society for Optics and Photonics, 2008.

[20] Bill Buxton. *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann, 2010.

[21] Reinhard Diestel. *Graphentheory*. Springer, 2000.

[22] JAVIER I ZARATIEGUI FERNANDEZ. *INTELLIGENT DESIGN OBJECTS APPLIED TO THE SPATIAL ALLOCATION PROBLEM*. PhD thesis, MIDDLE EAST TECHNICAL UNIVERSITY, 2014.

[23] Ian Fette. The websocket protocol. 2011.

[24] Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk. Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? In *Proceedings of the*

SIGCHI conference on Human Factors in Computing Systems, pages 345–352. ACM, 2000.

[25] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[26] John D Gould and Clayton Lewis. Designing for usability: key principles and what designers think. *Communications of the ACM*, 28(3):300–311, 1985.

[27] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[28] Jiang Guo. Backpropagation through time. *Unpubl. ms., Harbin Institute of Technology*, 2013.

[29] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE, 1989.

[30] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Tao-Kuang Huang, Larry O Degelman, and Terry R Larsen. A visualization model for computerized energy evaluation during the conceptual design stage (energraph). 1992.

[33] Timo Jokela, Netta Iivari, Juha Matero, and Minna Karukka. The standard of user-centered design and the standard definition of usability: analyzing iso 13407 against iso 9241-11. In *Proceedings of the Latin American conference on Human-computer interaction*, pages 53–60. ACM, 2003.

[34] Olga Kulyk, Robert Kosara, Jaime Urquiza, and Ingo Wassink. Human-centered aspects. *Human-centered visualization environments*, pages 13–75, 2007.

[35] Christoph Langenhan. *Datenmanagement in der Architektur*. Dissertation, Technische Universitt Mnchen, Mchen, 2017.

[36] Jia-Her Lee and Yu-Tung Liu. Modelling mondrian's design processes and their architectural associations using multilayer neural networks. 1998.

[37] Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. In *ACM Transactions on Graphics (TOG)*, volume 29, page 181. ACM, 2010.

[38] Justin J Miller. Graph database applications and concepts with neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, volume 2324, page 36, 2013.

[39] Lance A Miller and John C Thomas. Behavioral issues in the use of interactive systems. *International Journal of Man-Machine Studies*, 9(5):509–536, 1977.

[40] Herbert Moelle. Rechnergestützte planungsprozesse der entwurfsphasen des architekten auf basis semantischer modelle. 2006.

[41] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[42] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[43] Siu-hang Or, Kin-Hong Wong, Ying-kin Yu, Michael Ming-yuan Chang, and H Kong. Highly automatic approach to architectural floorplan image understanding & model generation. *Pattern Recognition*, pages 25–32, 2005.

[44] John R Ragazzini and Lotfi A Zadeh. The analysis of sampled-data systems. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, 71(5):225–234, 1952.

[45] Jon D Reid. *jQuery Mobile*. O'Reilly Germany, 2011.

[46] Katharina Richter. *Augmenting designers' memory: case based reasoning in der Architektur*. Logos Verlag Berlin GmbH, 2011.

[47] Qamer Uddin Sabri, Johannes Bayer, Viktor Ayzenshtadt, Syed Saqib Bukhari, Klaus-Dieter Althoff, and Andreas Dengel. Semantic pattern-based retrieval of architectural floor plans with case-based and graph-based searching techniques and their evaluation and visualization.

[48] Krystian Samp and Stefan Decker. Supporting menu design with radial layouts. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 155–162. ACM, 2010.

[49] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[50] Brian Shackel. Usability-context, framework, definition, design and evaluation. *Human factors for informatics usability*, pages 21–37, 1991.

[51] Sidney L Smith and Jane N Mosier. *Guidelines for designing user interface software*. Mitre Corporation Bedford, MA, 1986.

[52] Heather Williamson. *XML: The complete reference*. McGraw-Hill Professional, 2001.