Information Modelling and Knowledge Bases XXXIV
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA220493

Heuristics for Spatial Data Descriptions in a Multi-Agent System

Marek MENŠÍK [a], Adam ALBERT [a], Petr RAPANT [a] and Tomáš MICHALOVSKÝ [b]
[a] *VSB - Technical University of Ostrava, 17. listopadu 15, 708 00 Ostrava, Czech Republic*
[b] *Silesian University in Opava, Berzučovo nám. 13, 746 01 Opava, Czech Republic*

Abstract. Navigation and an agent's map representation in a multi-agent system become problematic when agents are situated in complex environments such as the real world. Challenging modifiability of maps, long updating period, resource-demanding data collection makes it difficult for agents to keep pace with rather quickly expanding cities. This study presents the first steps to a possible solution by exploiting natural language processing and symbolic methods of supervised machine learning. An adjusted algorithm processes formalized descriptions of one's journey to produce a description of the journey. The explication is represented employing Transparent Intensional Logic. A combination of several explications might be used as a representation of spatial data, which may help the agents to navigate. Results of the study showed that it is possible to obtain a topological representation of a map using natural language descriptions. Collecting spatial data from spoken language may accelerate updating and creation of maps, which would result in up-to-date information for the agents obtained at a rather low cost.

Keywords. Machine learning, Refinement, Generalization, Specialization, Hypothesis, Heuristics

1. Introduction

Multi-agent systems are commonly embedded in the space. In order to operate, they need to explore the structure of this space and learn to navigate within it. They need to build up a knowledge base representing the locations in the space, the possibilities of moving between them, and possibly other accompanying information. For this purpose, it is possible to use the means of supervised machine learning, specifically natural language processing, which is an interdisciplinary discipline involving linguistics, logic and computer science [1]. This paper aims to describe the application of machine learning methods in agents' learning paths in the city. The system uses the formal knowledge representation by means of Transparent Intensional Logic (TIL) constructions. We deal with learning based on the representation of agents' journey.

A similar algorithm was exploited in several papers that obtained explications used for several applications. The seeking relevant information source from many textual sources

was introduced in [2]. Using Formal Conceptual Analysis (FCA) and Association rules analysis, seeking additional appropriate textual sources was introduced in [3], [4]. Improvement of recommendation method using Iceberg concept lattices was introduced in [5]. Explications obtained with a combination of FCA were used in [6] for seeking the most appropriate concept. In [7] and [1], authors processed natural language sentences containing information about a particular concept.

Agents' journey through the city can be described as an ordered set of sentences. Each sentence describes one whole part of the journey, usually containing information about who, how, from where, towards where, along with what and for how long the agent moved. For simplicity, we will assume that the consecutive individual parts of the path, described by particular sentences, are mutually connected. Therefore, some of the missing information from the following sentence can be taken from the previous sentence. Once a path representation is created based on the available description (path introduction), it can be further updated based on new information obtained from other agents' reports (path update).

The paper is structured as follows. Chapter 2 summarises supervised machine learning and Chapter 3 describes inductive heuristics. Chapter 4 demonstrates inductive heuristics functionality on some examples. Chapter 5 contains conclusions and future work descriptions.

2. Supervised Machine Learning

Supervised machine learning is a task of learning functional dependency based on observing classified positive or negative examples. Supervisor to an agent provides those examples. Examples are described by a set of input and output attributes. There is an unknown functional dependency f between values of those two sets. Observing the values of input and output attributes of examples, the agent builds his functional dependency h called the hypothesis. The hypothesis should approximate the original unknown function dependency f. Using this hypothesis, the agent should be able to predict the value of output attributes based on values of input attributes on unseen examples.

For example, conditions for receiving a loan by a bank can be described by input attributes *employment*, *salary*, *age*, *indebtedness*, and *health condition* of an applicant. The risk of providing a loan to the applicant is the output attribute.

The hypothesis can be verified by a set of test examples where only values of input attributes are known to the agent. If the hypothesis predicts the values of output attributes as the original dependency f on test examples, the hypothesis is correct. More about supervised machine learning can be found in [8], [9], [10].

In the following sections, we briefly summarise the supervised machine learning algorithm and its adjustments to our previous works leading to the current stage of development.

2.1. Building a Concept

In [7], we exploited Patrick Winston's symbolic algorithm [11] to obtain a hypothesis of a concept classifying unknown examples described by the language of TIL-script construction. TIL-script is a computation variant of Transparent Intensional Logic (TIL). TIL is a partial, typed hyperintensional lambda calculus with procedural as instead of set-theoretical denotational semantics.[1]

The algorithm's goal was to find a general concept specifying the property of being an *arch*.[2] *Arches* are built from blocks of different colors, positions, and shapes. The resulting molecular concept should be general enough that the agent should be able to identify those individuals that belong to the class of arches and at the same time specific enough to exclude those individuals that do not belong to this class.

The algorithm built a general concept using two heuristic methods, namely *Generalization* and *Specialization*. Both methods contain heuristic functions that manipulate the symbolic representation of the hypothesis.

Positive examples are processed by the Generalization that adjusts the hypothesis, i.e. molecular construction, therefore more objects can be correctly identified as the concept described by the hypothesis.

Negative examples are processed by Specialization that serves to distinguish the output concept from similar ones. For instance, a wooden horse can serve as a negative example to the concept of the horse, because a wooden horse is not a horse; instead, it is a toy horse though it may look like a genuine living horse.

2.2. Refinement of a Concept

In [1] we adjusted the [7] algorithm for natural language processing. The goal was to create explications of atomic concepts using natural language sentences. Carnapian explication is a refinement process of an inaccurate or vague expression into an adequately accurate one. For example, the atomic concept of *a Dog* might have an explication that *A dog is a domesticated carnivore*. Therefore, the goal of an agent is to discover the best refinement of the learned simple concept of an object O, i.e. a molecular closed construction that produces the same object. Moreover, this molecular concept should specify the requisites of the object O as much as possible so that it excludes other similar concepts.

Original Winston's algorithm uses examples with complete information about the concept. The agent's hypothesis differs from the presented example in one significant difference, and the hypothesis is adjusted by the machine learning algorithm based on that one significant difference.

However, in [1] examples are in the form of natural language sentences formalized into the language of TIL-script constructions. Unlike Winston's examples, natural language sentences do not carry complete information instead, they contain new (partial) infor-

[1] TIL has been introduced in numerous papers and two books, see, for instance [12], [13], [14], [15], [16], [17].

[2] A concept is a closed construction in its normal form in the notion of TIL.

mation about the explicated object. Hence, examples differ in more than one significant difference compared to the hypothesis.[3] It was necessary to modify the algorithm.

We have introduced a new heuristic method called *Refinement*. This method contains one heuristic function called *Concept introduction*, and this function inserts new information about the object to be explicated into the molecular construction as its new constituent.

2.3. Network Building

In this paper, we are dealing with a new challenge. In [7] and [1] we processed natural language sentences mentioning a particular concept, and those sentences contained partial information about a particular concept. We also deal with natural language sentences formalized into TIL-script constructions but as different examples. Examples now contain pieces of information about an agent's journey, and a new method of processing them must be implemented. At this time, we are not refining a simple concept; instead we build an agent's journey from its description in the form of TIL-script constructions, i.e. we build molecular construction describing the agent's journey.

To connect pieces of information contained in examples, we exploit the class of motion verbs, for example, *to go to walk*, *to cross*, *to turn*, *to come*, etc., and valency frames. Valency might be seen as the capacity a verb has for combining with particular patterns of other sentence constituents.[4] Valency frames provide information on which kind of complement is a valency bound with a particular verb in a particular sentence. For example, a verb *to come* might by valency bond in a sentence with complements such as an *actor* (who is walking), *directions* (from, where to, which way), *manner* (sped of the walk, the direction of the walk), *extent* (how far).

For one's journey analysis we are using the following *functors* that indicate the types of syntactic-semantic relationship between a verb and its complement that might occur in valency frames:

- Actor (ACT): ***Paul*** *is riding a bike.*
- Direction - from (DIR1) : *He came* ***from woods*** *soaked wet.*
- Direction - which way (DIR2): *He went to the neighboring village* ***through the forest****.*
- Direction - where (DIR3): *He went* ***to the neighboring village*** *through the forest.*
- Manner (MANN): *He treated her* ***kindly***.
- Extent (EXT): *Dad measured* ***2 meters***.

The sentence *"Tom walks quickly from home to school."* might be analyzed by exploiting the valency frame of the verb *to walk* as follows:

- ACT: Tom
- DIR1: home

[3] For details, see [11].
[4] For details, see [18].

- DIR3: school
- MANN: quickly

Using valency frames and information that follows from them, we might obtain a formal description of one's journey by description in natural language formalized in the expressive language of TIL.[5]

$$\lambda w \lambda t\ [[{}'ACT_{wt}\ {}'Tom\ {}'walk] \wedge [{}'DIR1_{wt}\ {}'home\ {}'walk] \wedge [{}'DIR3_{wt}\ {}'school\ {}'walk] \wedge [{}'MANN_{wt}\ {}'quickly\ {}'walk]]$$

Types:
DIR1, DIR3/$(o\pi\nu)_{\tau\omega}$*; ACT/* $(o\iota\nu)_{\tau\omega}$*; MANN/* $(o\alpha\nu)_{\tau\omega}$*; Tom/* ι*; home, school /*π*; walk/*ν, where π is type of places; ν is type of the activity denoted by a verb.

As algorithms introduced in [7] and [1], we can also describe adjusted algorithms using a general framework for symbolic methods of supervised machine learning. This general framework consists objectives, training data, data representation, and a module for manipulation with the symbolic representation.

- ***Objectives***: Our algorithm produces hypotheses in the form of molecular TIL-Script construction.
- ***Training data***: Training data are natural language sentences that describe one's journey. Hence, they contain information about traveller's names, directions, attributes of the traveling such as speed, distances, changes in directions, etc.
- ***Data representation***: Same as previously mentioned algorithms, this algorithm also exploits the TIL. For computational purpose, training data are formalized in TIL-script language.
- ***Module for manipulation with the symbolic representation***: This module contains three heuristic methods that manipulate the hypothesis of one's journey based on training data examples. The methods are *Generalization*, *Specialization*, and *Refinement*. This paper is dedicated to the *Refinement* method in which we define two new heuristic functions: *Path Introduction* and *Path Update*.

  Path Introduction inserts new pieces path pieces into the molecular description, thus extending the path description. Using *Path Update*, the path description is altered with a possibly more precise description. For example, one might describe his journey briefly, and the brief description serves us as a hypothesis of the journey. Another one might describe the same journey more in detail. *Path Update* will alter the hypothesis so the description of the journey might be as detailed and accurate as possible.

[5]For the sake of readability we will use just TIL language to display examples. Our computations are executed over TIL-Script constructions.

3. Inductive Heuristics

A journey description is obtained by exploiting the valency structure of motion class verbs. We divide the journey description into two kinds of information; the core information about directions, distance, and places we call the *network* and the description of surroundings on the journey. The description of surroundings is not a topic of this paper and it will be a subject of our future work. In this paper, we have focused on the first part; the creation of the *network*.

The journey is a sequence of places that one might visit, complemented with additional information such as an actor who took the journey, the distance it took the actor to move from one place to the other, etc. Valency frames of motion class verbs identify that information. Places of the journey are identified by *direction's* functors (DIR1, DIR2, DIR3). Change in the movement direction change of speed is identified by the *manner* functor (MANN). The one who took the journey is identified by the *actor* functor (ACT). The *extent* functor (EXT) identifies the distance between two places.

Places are the essential information in the *network*; therefore, we call *nodes* constituents of *direction's* functors. Places might be connected via valency to the motion verb. Direction functors (DIR1 - from, DIR3 where) determine the direction. Motion verb together with other functors (such as ACT, MANN, EXT) we call *edge*.

Definition 1 (*node, edge*)**.** Let V be a *motion verb*, let $S = \{B | ('DIR1 \text{ or } 'DIR3) \text{ and } V \text{ are constituents of B } \}$ and let $D^V = \{C | V \text{ is a constituent of C } \} \setminus S$ then D^V is a set of ***edges*** and S is a set of ***nodes***.

Definition 2 (*place, functor, value*)**.** Let $[\alpha\ x\ v]$ be a *node* and let $[\beta\ y\ v_1]$ be an *edge*. Then x is a ***place***, α, β are ***functors***, and y, v, v_1 are ***values***.

A simple sentence might connect two places via verb valency with additional information. For example, let us have a sentence *"John went from X to Y."*, formalized as:

$$\lambda w \lambda t\ [['ACT_{wt}\ 'John\ 'went] \wedge ['DIR1_{wt}\ 'X\ 'went] \wedge ['DIR3_{wt}\ 'Y\ 'went]] \qquad (1)$$

By definition 1, $['DIR1_{wt}\ 'X\ 'went]$ and $['DIR3_{wt}\ 'Y\ 'went]$ are *nodes*. The rest, $['ACT_{wt}\ 'John\ 'went]$, is an edge. By definition 2, X and Y are *places*; they are constituents of *nodes*. We can define a *connection* using nodes and edges, thus forming a *network*.

Definition 3 (*connection*)**.** Inductive definition:

1. Let α be an *edge* then α is a *connection (atomic)*.
2. Let α, γ be *connections*, let β be *node* then $\alpha \rightarrow \beta \rightarrow \gamma$ is a ***connection***.
3. Only structures in 1 and 2 are *connections*.

Remark: Connection is a transition between nodes.

Definition 4 (*path*)**.** Let α and γ be *nodes* and let β be a *connection* then $\alpha \rightarrow \beta \rightarrow \gamma$ be a *path*.

Definition 5 (*same path*)**.** Let $\alpha \rightarrow \beta \rightarrow \gamma$ and $\alpha \rightarrow \delta \rightarrow \gamma$ be *paths* in model and positive example respectively, let $\delta = \varepsilon \rightarrow ... \rightarrow \omega$ then if $\lambda x['DIR2_{wt}\ x\ y]$ in β (model) $= \lambda y['DIR2_{wt}\ y\ z]$ in δ (positive example) then both paths represent the ***same path***.

Definition 6 (*network*)**.** A *network* is a set of all *paths*.

3.1. Path Preprocessing

This section describes the essential part of our algorithm. *Path Introduction* and *Path Update* can operate only over the *paths*. Therefore, input constructions representing a route description must be converted to constructions representing a *path* by definition 4.

Following Figure 1 represents *path* preprocessing. The algorithm consists of decision parts, namely *Path, Sen1, DIR1, DIR3*, and procedures.
The decision represented by *Path* tests whether the input description represents a *path*; *Sen1* check if the construction represents the first sentence; *DIR*1 and *DIR*3 check whether there are constituents $'DIR1$ and $'DIR3$ in the input construction or not respectively.
Procedures $D1^S \Leftarrow D3^{S-1}$ and $D3 \Leftarrow Dummy$ inserts *nodes* to produce *path* by the definition.[6]

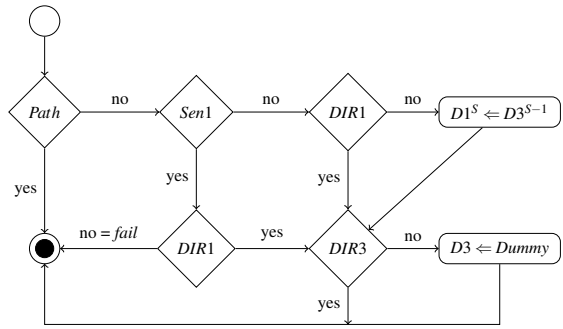

Figure 1. *Path Preprocessing* flow diagram

For a better reader's understanding, we will describe the functionality of the particular parts of the preprocessing in more detail here.

Decisions
We have three basic types of decision-making. The first one checks whether the input is a *path*-describing construction (*Path*). The second one checks whether the input represents the first sentence (*Sen1*). The third one (*DIR*1, *DIR*3) verifies that the input description contains constituents representing the beginning and the end of the *path*.

Path
If an *input* is of the form $\alpha \rightarrow \beta \rightarrow \gamma$ where α and γ are *nodes* and β is a *connection* then input description is a *path*. If not so then proceed to decision *Sen*1.

*Sen*1 + *DIR*1
At this point, it is necessary to check whether the *input* represents the first sentence. If so, it is crucial to know where the *path* will begin. Therefore, if there is no information about *D*1, we cannot start the process and it is necessary to ask the user to add this crucial information.[7] If *D*1 is included then we can proceed to *DIR*3.

[6] $D3^{S-1}$ represents *node* containing constituent $'DIR3$ from previous sentence.
[7] *Di* represents node $[{}'DIRi_{wt}\ x\ y]$

*DIR*3
This decision part checks whether our input contains the information about the end or heading of a specified direction. If there is no such information we trigger the $D3 \Leftarrow$ *Dummy* procedure which introduces the given description node with *dummy* $D3$.

*DIR*1
This condition verifies that the description includes the direction's origin. If it does, it is proceeding to *DIR*3. If $D1$ is not included in the description and not the firs sentence, then $D1$ is added to the description. The new $D1$ is obtained from node $D3$ of the previous sentence.

Algorithm represented by figure 2:

Let's have a sequence of constructions as the input: $(S_1, S_2, \ldots, S_n)$, $I = \{1, \ldots, n\}$.

start: For each $i \in I$ do[8]

(a) IF $i = 1$ then *dirs1*
(b) IF *'DIR*1 in Si then *dir3*
(c) *introduce* $[{}'DIR1_{wt}$ *x verb*$]$ into S_i where *x, the verb* is taken from $D3^{S_{i-1}}$
(d) *dir3*

dir3: IF *'DIR*3 in Si then *end*

(a) *introduce* $[{}'DIR3_{wt}$ *'Dummy verb*$]$ into S_i, where the *verb* is taken from S_i
(b) *end*

dirs1: IF *not*(*'DIR*1 in S_1) then *fail*

(a) *dir3*

end: return S_i

3.2. Path Introduction / Path Update

Several heuristics must be applied to create the path *network*. Since we process spatial data, a new heuristic had to be introduced to update the path. The heuristic for introducing a new path remains unchanged. [9] Deciding which heuristic to run is based on path comparison. If the paths are the same, the *Update Path* heuristic is triggered.

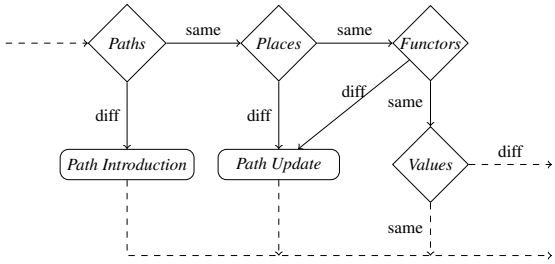

Figure 2. *Refinement heuristics*

[8] *start*/(c) represents procedure $D1^S \Leftarrow D3^{S-1}$
[9] *Path introduction* correspond to the heuristic *Concept introduction* in [1]

4. Inductive Heuristics' Functionality Examples

Assume the following situation: We have two different journey's descriptions. One is from Tom's point of view and the other is from John's point of view. The following sentence describes Tom's perspective:

- *"Tom is walking down A street from B to C."*

TIL *explication* mapping the sentence is:

$$\lambda w \lambda t [[{}'ACT_{wt}\ {}'Tom\ {}'walking] \wedge [{}'DIR1_{wt}\ {}'B\ {}'walking] \\ \wedge [{}'DIR2_{wt}\ {}'A\ {}'walking] \wedge [{}'DIR3_{wt}\ {}'C\ {}'walking]] \tag{TR}$$

These sentences describe John's perspective:

- *"John came to E from B on street A."*
- *"After 100m John came to F on A street"*
- *"And after another 100m John came to C on street A."*

TIL descriptions:

$$\lambda w \lambda t [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'B\ {}'came] \\ \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'E\ {}'came]] \tag{J_1}$$

$$\lambda w \lambda t [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \\ \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'F\ {}'came]] \tag{J_2}$$

$$\lambda w \lambda t [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \\ \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'C\ {}'came]] \tag{J_3}$$

The description from John's sentences will be as follows:

$$\lambda w \lambda t\ [[[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'B\ {}'came] \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \\ \wedge [{}'DIR3_{wt}\ {}'E\ {}'came]] \wedge [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'E\ {}'came] \\ \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'F\ {}'came]] \\ \wedge [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'F\ {}'came] \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \\ \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'C\ {}'came]]] \tag{JR}$$

To obtain the final *journey explication* which is in our case (JR) we need to go through several steps which follow:

4.1. Path Preprocessing

In order to start the actual explication process, it is required to prepare the input structures first. All TIL constructions, namely TR, J_1, J_2, J_3, need to meet the *path* conditions.

- TR $\rightarrow$ passed
- J_1 $\rightarrow$ passed
- J_2 $\rightarrow$ missing node $D1$ $\rightarrow$ run ($D1^{J_2} \Leftarrow D3^{J_1}$)

$$\lambda w \lambda t\ [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'E\ {}'came] \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'F\ {}'came]] \quad (J_2^*)$$

- J_3 $\rightarrow$ missing node $D1$ $\rightarrow$ run ($D1^{J_3} \Leftarrow D3^{J_2}$)

$$\lambda w \lambda t\ [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'F\ {}'came] \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'C\ {}'came]] \quad (J_3^*)$$

We have prepared all the necessary inputs to run the heuristics at the end of this process The following heuristics in sections 4.2 and 4.3 are called separately.

4.2. Path Introduction

Because the descriptions TR and J_1 meet the *path* conditions we do not need to preprocess them. The given construction TR is already an *explication* of a given journey. The construction J_1 represents just a part of a journey; therefore it needs to be used as an input for *heuristics*. Constructions J_2 and J_3 had to be modified into single path descriptions by the *path preprocessing* where we obtained constructions J_2^* and J_3^* . All these constructions (J_1, J_2^*, J_3^*) are merged, represented as John's journey *explication* JR.
For these reasons, we apply the *Path Introduction* heuristic.

The *Path introduction* proceeds as follows:

1. The first construction, J_1, becomes a *model*.
   (a) The second construction, J_2^*, is a positive example.
   (b) Introduce J_2^* into a *model* J_1.

$$\lambda w \lambda t [[[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'B\ {}'came] \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'E\ {}'came]] \wedge [[{}'ACT_{wt}\ {}'John\ {}'came] \wedge [{}'DIR1_{wt}\ {}'E\ {}'came] \wedge [{}'EXT_{wt}\ {}'100\ {}'came] \wedge [{}'DIR2_{wt}\ {}'A\ {}'came] \wedge [{}'DIR3_{wt}\ {}'F\ {}'came]]] \quad (J_{12})$$

2. J_{12} is a new *model*.
   (a) The construction J_3^* is a positive example.
   (b) Introduce J_3^* into a *model* J_{12}.

$$\begin{aligned}\lambda w\lambda t[[[['ACT_{wt}\ 'John\ 'came] \wedge ['DIR1_{wt}\ 'B\ 'came] \wedge ['DIR2_{wt}\ 'A\ 'came]\\ \wedge['DIR3_{wt}\ 'E\ 'came]] \wedge [['ACT_{wt}\ 'John\ 'came] \wedge ['DIR1_{wt}\ 'E\ 'came]\\ \wedge['EXT_{wt}\ '100\ 'came] \wedge ['DIR2_{wt}\ 'A\ 'came] \wedge ['DIR3_{wt}\ 'F\ 'came]] \quad \text{(JR)}\\ \wedge[['ACT_{wt}\ 'John\ 'came] \wedge ['DIR1_{wt}\ 'F\ 'came] \wedge ['EXT_{wt}\ '100\ 'came]\\ \wedge['DIR2_{wt}\ 'A\ 'came] \wedge ['DIR3_{wt}\ 'C\ 'came]]]\end{aligned}$$

After obtaining all the journeys' explications TR and JR, we follow a process that compares the explications. They describe the *same path*, triggering the (*Path Update)* heuristic.

4.3. Path Update

1. The first explication TR is becoming a *model*. This *model* contains all parts necessary for defining a *path*:
   Nodes:

   a $['DIR1_{wt}\ 'B\ 'walking]$

   b $['DIR3_{wt}\ 'C\ 'walking]\}$

   Edges:

   α $\{['ACT_{wt}\ 'Tom\ 'walking], ['DIR2_{wt}\ 'A\ 'walking]\}$

2. Second explication JR is a positive example. This example represents *path* as well.
   Nodes:

   c $['DIR1_{wt}\ 'B\ 'came]$

   d $['DIR3_{wt}\ 'E\ 'came]$

   e $['DIR1_{wt}\ 'E\ 'came]$

   f $['DIR3_{wt}\ 'F\ 'came]$

   g $['DIR1_{wt}\ 'F\ 'came]$

   h $['DIR3_{wt}\ 'C\ 'came]$

   Edges:

   β $\{['ACT_{wt}\ 'John\ 'came], ['DIR2_{wt}\ 'A\ 'came]\}$

   γ $\{['ACT_{wt}\ 'John\ 'came], ['EXT_{wt}\ '100\ 'came], ['DIR2_{wt}\ 'A\ 'came]\}$

   δ $\{['ACT_{wt}\ 'John\ 'came], ['EXT_{wt}\ '100\ 'came], ['DIR2_{wt}\ 'A\ 'came]\}$

3. We are checking the possibility that they are the *same path*.

   (a) Structures:
   Model: $\{a\} \rightarrow \alpha \rightarrow \{b\}$
   Example: $\{c\} \rightarrow \beta \rightarrow \{d,e\} \rightarrow \gamma \rightarrow \{f,g\} \rightarrow \delta \rightarrow \{h\}$

(b) Checking first and last places in paths
First-model {a}: $\lambda x['DIR1_{wt}\ x\ 'walking] = \{'B\}$
First-example {c}: $\lambda x['DIR1_{wt}\ x\ 'came] = \{'B\}$
Last-model {b}: $\lambda x['DIR3_{wt}\ x\ 'walking] = \{'C\}$
Last-example {h}: $\lambda x['DIR3_{wt}\ x\ 'came] = \{'C\}$

(c) Checking DIR2 equality:
Model: $\lambda x['DIR2_{wt}\ x\ 'walking] = \{'A\}$
Example: $\lambda x['DIR2_{wt}\ x\ 'came] = \{'A\}$
$\Rightarrow$ *same path*

4. Replacing connections
Replace *model's* connection by *example's* connection:
$\alpha \Rightarrow (\beta \rightarrow \{d,e\} \rightarrow \gamma \rightarrow \{f,g\} \rightarrow \delta)$

5. The final updated *Path*:
$\{a\} \rightarrow \beta \rightarrow \{d,e\} \rightarrow \gamma \rightarrow \{f,g\} \rightarrow \delta \rightarrow \{b\}$

Since we have already processed all the explications, triggering the heuristics is complete. The resulting explication covers both journeys, resulting in the desired *network*.

5. Conclusion

This paper deals with building representation of space using descriptions of journeys. Descriptions are obtained from formalized natural language sentences processed by a supervised machine learning algorithm. They might be combined into a network describing space. Such a network can be used as a navigation tool in complex environments of multi-agent systems, e.g., cities. Creating and modifying explications are realized by applying heuristic functions of the adjusted algorithm, which we introduced in this paper. The paper's novelty is the processing of the journey's description exploiting specific motion verbs' valency frames in the descriptions.

The developed procedure of converting the textual description of possible journeys around the city into a formalized form, enriched with much accompanying information, makes it possible to create a more detailed representation of a given space gradually. It might be possible to develop a city orientation plan from this representation in the future. The next step will be to link this orientation plan with the existing map in GIS and enrich it with information obtained from agents.

Acknowledgements

This research has been supported by Grant of SGS No. SP2022/123, VŠB - Technical University of Ostrava, Czech Republic, "Application of Formal Methods in Knowledge Modelling and Software Engineering V" and by CZ.02.2.69/0.0/0.0/18_054/0014696 Development of R&D capacities of the Silesian University in Opava and also supported under the Student Funding Scheme, project SGS/8/2022.

References

[1] Menšík, M., Duží, M., Albert, A., Patschka, V., Pajr, M. (2019): Refining concepts by machine learning.*Computación y Sistemas*, Vol. 23, No. 3, 2019, pp. 943–958, doi: 10.13053/CyS-23-3-3242

[2] Menšík, M., Duží, M., Albert, A., Patschka, V., Pajr, M. (2019): Seeking relevant information sources. In Informatics'2019, *IEEE 15th International Scientific Conference on In-formatics*, Poprad, Slovakia, pp. 271-276.

[3] Menšík, M., Albert, A., Patschka, V. (2020): Using FCA for Seeking Relevant Information Source. In *RASLAN 2020*, Brno: Tribun EU, 2020, 144 p. ISBN 978-80-263-1600-8, ISSN 2336-4289.

[4] Albert, A., Duží, M., Menšík, M., Pajr, M., Patschka, V. (2021): Search for Appropriate Textual Information Sources. In *Frontiers in Artificial Intelligence and Applications*, vol. 333: Information Modelling and Knowledge Bases XXXII, B. Thalheim, M. Tropmann-Frick, H. Jaakkola,N. Yoshida, Y. Kiyoki (eds.), pp. 227-246, Amsterdam: IOS Press, doi: 10.3233/FAIA200832

[5] Menšík, M., Albert, A., Patschka, V. a Pajr, M. Improvement of Searching for Appropriate Textual Information Sources Using Association Rules and FCA. In: Information Modelling and Knowledge Bases XXXIII. Amsterdam: IOS Press, 2022, 2022-01-14. Frontiers in Artificial Intelligence and Applications. ISBN 9781643682426. ISSN 978-1-64368-242-6. Dostupné z: doi:10.3233/FAIA210487

[6] Menšík, M., Albert A., Michalovský T. Using FCA and Concept Explications for Finding an Appropriate Concept. In: Proceedings of the Fifteenth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2021. Brno: Tribun EU, 2021, s. 49-60. ISBN 978-80-263-1670-1. ISSN 2336-4289.

[7] Menšík, M., Duží, M., Albert, A., Patschka, V., Pajr, M., "Machine learning using TIL". In Frontiers in Artificial Intelligence and Applications, Amsterdam: IOS Press, Vol. 321, pp. 344-362, DOI: 10.3233/FAIA200024

[8] Russell, S.J.,Norvig, P.: "Artificial intelligence: a modern approach.", 2nd ed. Harlow, Pearson Education, 2014. ISBN 978-1-29202-420-2.

[9] Mitchell, T.M.: "Machine learning", New York: McGraw-Hill, 1997. ISBN 00-704-2807-7.

[10] Poole, D.L., Mackworth, A.K.: "Artificial intelligence: foundations of computational agents. 2nd pub.", Cambridge: Cambridge University Press, 2010, ISBN 978-0-521-51900-7.

[11] Winston, P. H.: "Artificial Intelligence". 3rd ed., Mass.: Addison-Wesley Pub. Co., 1992. ISBN 02-015-3377-4.

[12] P. Tichý, "Collected Papers in Logic and Philosophy", eds. V. Svoboda, B. Jespersen, C. Cheyne. Prague: Filosofia, Czech Academy of Sciences, and Dunedin: University of Otago Press, 2004.

[13] Duží, M., Jespersen, B., Materna, P. (2010): Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Transparent Intensional Logic. Berlin: Springer.

[14] Číhalová, M., Duží, M., Menšík, M. (2014): Logical specification of processes. Frontiers in Artificial Intelligence and Applications, vol. 260: Information Modelling and Knowledge Bases XXV, IOS Press, 45-63.

[15] Duží, M., Číhalová, M., Menšík, M. (2011): Communication in a multi-agent system based on Transparent Intensional Logic. In Mendel 2011, Matousek, R. (ed.), Brno: VUT, pp. 477-485.

[16] Menšík, M., Duží, M., Kermaschek, J. (2018): The role of beta conversion in functional programming. Frontiers in Artificial Intelligence and Applications, vol. 301: Information Modelling and Knowledge Bases XXIX, pp. 280-298, Amsterdam: IOS Press, DOI 10.3233/978-1-61499-834-1-280

[17] Menšík, M., Duží, M., Pajr, M, Patschka, V. (2018): Natural Deduction System in the TIL-Script Language. In Proceedings of the 28th International Conference on Information Modelling and Knowledge Bases, EJC 2018, Tatiana Endrjukaite, Hannu Jaakkola, Bernhard Thalheim and Yasushi Kiyoki (eds.), pp. 220-236, ISBN 978-9984-818-89-4.

[18] Fischer, K., Ágel, V. Dependency grammar and valency theory. In: The Oxford handbook of linguistic analysis; 2010, p. 223-255.