Baltic J. Modern Computing, Vol. 9 (2021), No. 2, 195-209
https://doi.org/10.22364/bjmc.2021.9.2.04

Structuring and Controlling the Knowledge for the Software User Support

Juris RĀTS, Inguna PEDE

RIX Technologies
Blaumaņa 5a-3, Rīga, LV-1011, Latvia

juris.rats@rixtech.lv, inguna.pede@rixtech.lv

Abstract. The research aims to create a smart user Support Assistant LAVA – a solution providing context sensitive and user experience aware support to a user of a software product. The solution is based on machine learning and search in a two-level distributed knowledge store. LAVA model focuses on providing user on a mouse click with a content relevant to a current context. LAVA uses context transferred from the supported product to instantly show a FAQ list of items used and/or positively rated by other users in the same context. Full-text search and a topic hierarchy are provided as well to cover as many of cases as possible given the content available currently in the Knowledge base. Main design ideas of the LAVA model are presented in the article including the calculation of ranges and creation of ranged context-sensitive FAQs, customizing the full-text search, organizing the topic hierarchy and integrating with the Service desk. The article covers as well the main considerations on how the LAVA model should be implemented in a successful solution, and a description of method to evaluate the model performance.

Keywords: user support, case-based reasoning, context-sensitive knowledge, machine-learning, full-text search, knowledge transfer.

1. Introduction

The aim of our research is to create a framework for a smart user Support Assistant LAVA (Latvian acronym) - an automated service desk solution to support a user of a software product (Product). The main benefits of our solution are:

- improved user training based on an easily accessible context-sensitive Knowledge store of answers to routine questions of the users;
- reduced amount of time spent by support staff on answering routine questions as the staff adds the related knowledge to the Knowledge store immediately accessible for other users.

As stated in (Trujillo, a) the real problem with customer support automation lies with an over-reliance on technology to do the jobs best left for real, live people. The technology should be used for routine, recurrent tasks while leaving the more complicated, less frequent ones to the human. In other words - humans should “do things that don’t scale” (Trujillo, a). We believe accordingly that humans should talk to each other while software should focus on things like user action logging and analysis, advanced search and support for knowledge management. Our solution therefore is not

going to pretend to be a chatbot but rather to provide a simple and user-friendly interface for access to context sensitive knowledge (Rats and Pede, 2020).

It is frequently stated that the aim of the support automation is to reduce or eliminate the need for human involvement when providing advice or assistance to user requests (Trujillo, a). This definition is a bit one-sided as it overlooks another source of the improvement – reducing the user need to create support requests in the first place. This can be achieved by better integration of knowledge management with user support and issue tickets management.

User support for a software product is mainly structured in two levels. The first level serves the users of a particular Product instance (e.g. the users of a specific institution) and is run by so-called power users. The power users handle routine problems raised at the Product instance level and escalate the service requests to the second level if necessary. The second level in turn is operated by the support staff of the Product provider.

This architecture has advantages and disadvantages. The most important disadvantage is that the second level support deals only with the end user problems escalated by power users. The rest (i.e. routine problems) stays locked inside the first level support unit and thus cannot be used by the second level support for the benefit of users of the rest of Product instances. We address this disadvantage in our solution by providing a mechanism of knowledge sharing between two levels of support.

2. Related work

Our research mainly focuses on defining self-service knowledge store based automated process for the software product users that would help them to find answers for routine questions and thus would reduce the requests to the Service desk.

One of the domains related to our research is recommender systems. There are basically two types of recommender systems, Content-based and Collaborative filtering (Raman, 2017). Content-based systems recommend similar items while collaborative filtering systems recommend items that relate to similar users (e.g. user bought also this). We use both approaches for our model.

Another technology related to our research is Case-based reasoning (CBR). CBR utilizes the specific knowledge of previously experienced problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems (Aamodt and Plaza, 1994).

At the highest level a CBR cycle may be described by the following four processes (Aamodt and Plaza, 1994):

- RETRIEVE the most similar case or cases;
- REUSE the information and knowledge in that case to solve the problem;
- REVISE the proposed solution;
- RETAIN the parts of this experience likely to be useful for future problem solving.

We use CBR process steps in our model as described in section 3.1.

2.1. FAQ and Knowledge Base software

A well-planned FAQ (Frequently Asked Questions) section is one of the tools that should help users to find solutions for themselves and thus to reduce the amount of service requests. A basic FAQ system comprises several groups of FAQ items and provides users with means to select an item group, browse and expand the questions to the relevant solution. This is enough if the FAQ contains up to a hundred or so items. The lack of more advanced means for information search restricts the ability of the FAQ system to grow the amount of the information. Either the less frequently used items must be removed from the FAQ or more advanced means must be introduced.

The search is the next information location means to consider. A number of FAQ solutions provide means for information search. Examples are Document360 (Graw, 2021), ProProofs Knowledge base (WEB, n) and HelpSite (WEB, j).

FAQ solutions may have a number of other features that mainly address needs of a support staff – like workflows and revision history (WEB, n) collaboration (WEB, c) and statistics (WEB, o). Still item grouping and search are the main end user means of the basic FAQ solutions.

Knowledge base functionality is provided as well by a number of helpdesk and ticketing systems.

2.2. Helpdesk and Ticketing solutions

Fontanella (2020) lists 14 freeware and commercial software and ticketing systems rated there as the top solutions for 2020. All but two (Agiloft and Spiceworks) of the listed systems feature knowledge base in some form.

Table 1 outlines knowledge base related features of the referred systems HubSpot's Service Hub (WEB, a), Jira Service Desk (WEB, d), Zoho Desk (WEB, q), C-Desk (WEB, f), SysAid (WEB, g), osTicket (WEB, e), ngDesk (WEB, k), Hesk (WEB, h), ManageEngine ServiceDesk Plus (WEB, l), HelpDeskZ (WEB, i), Web Help Desk by SolarWinds (WEB, p) and HelpSpot (WEB, m). Number in Topic Hierarchy column designates supported levels of topic hierarchy.

Knowledge bases of the listed systems as a rule feature article creation, topic hierarchy, search, tagging, embed media and user feedback. Most of them provide as well an access control.

Knowledge bases mainly are not context-sensitive though. The following interesting exceptions exist though:

- HubSpot’s service Hub provides a related articles feature; related articles are automatically chosen based on their relevance to the current article, that automatically adjusts based on article performance and visitor engagement;
- osTicket and ManageEngine ServiceDesk Plus both provide an option to link articles manually to help topics.

Table 1. Helpdesk and Ticketing systems.

Brand	Search	Topics	Categories, keywords, tags	Embed media	Feed-back	Access control
HubSpot's Service Hub	+	2	+	+	y/n	+
Jira Service Desk	+	+	+	+	comm.	+
Zoho Desk	+	4	+	+	+	+
C-Desk	+	+		+		
SysAid	+	+	+	+	y/n	
osTicket	+	+	+	+		
ngDesk	+	2	+	+	comm.	+
Hesk	+	1			rating	
ManageEngine ServiceDesk Plus	+	+	+	+	comm.	+
HelpDeskZ	+	+		+	rating	
Web Help Desk by SolarWinds	+	+		+	rating	+
HelpSpot	+	2	+			+

We did not find though any indication that the reviewed solutions (Helpdesk or FAQ) would use context information from the supported software product. Namely – a reference to a user interface item the user is activating a helpdesk system. The product interface page (form) user activates a help system should be significantly related to what a user is currently doing. Hence this can be used to determine what knowledge is most relevant and should be presented to the user upon the activation.

3. Proposed Solution

We follow in our model the recommendation that humans should “do things that don’t scale” (Trujillo, a). In other words – our model should be aimed at automating things

that scale. This is analyzed from a different perspective by Sutton (2020) stating that "general methods that leverage computation are ultimately the most effective" (when compared with methods trying to leverage human knowledge). As Sutton rightfully declares two methods outstand in their capability to scale - *search* and *learning*. These are the main methods we are willing to base our model upon.

The cornerstone design ideas for the LAVA model are:

- close mutual integration between the Knowledge store and the user service process;
- User is provided with the useful knowledge (context dependent FAQ) instantly as she activates LAVA; LAVA learns what is useful from user experience;
- Apart from the FAQ user is provided with means to search relevant knowledge – keyword search, topic browsing and navigating hyperlinks;
- User communicates with a service desk when fails to find useful knowledge; service desk may add new knowledge to the Knowledge store to help other users;
- Knowledge store is used both by a user to find solutions and by support staff to find similar cases that could be used for the user request in question;
- LAVA is integrated with the ticketing software in use; LAVA cares for communication with the user while the ticketing software handles ticket management, including SLA management, ticket routing etc.

LAVA uses five main data stores (Figure 1).

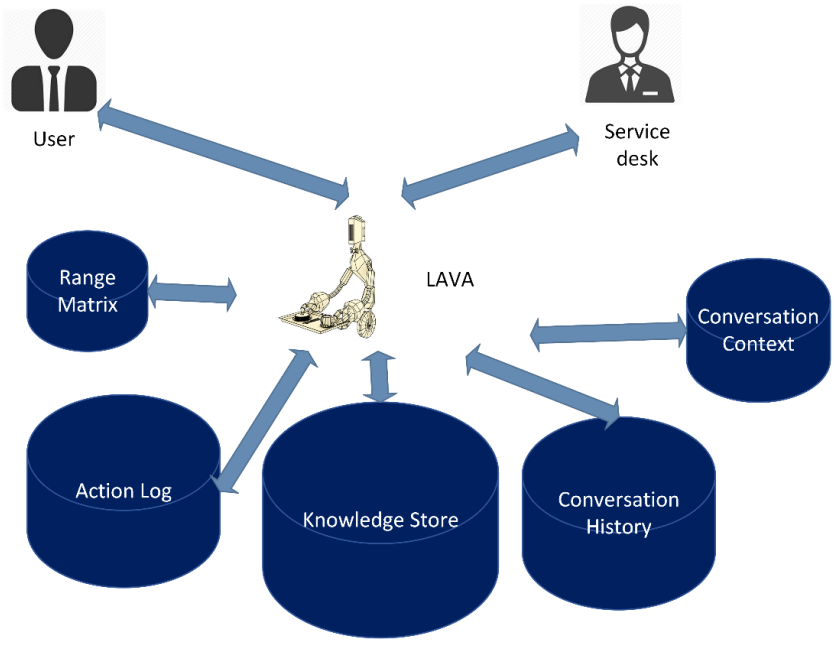

Figure 1. LAVA high level architecture (Rats and Pede, 2020).

Knowledge store is a set of knowledge items where each knowledge item comprises a number of attributes:

- question text;
- answer body (html text with hyperlinks to other knowledge items and media files (images, video);
- list of topics the item belongs to;
- list of contexts (UI item ids) of the Supported software product the item is relevant to;
- more attributes like related product module, version etc.

Action log saves user action data as a user opens the items of the Knowledge Store. Action log data are used to calculate the Range Matrix.

Range Matrix ranges knowledge items against the context (the Product UI items). Matrix is used to decide what knowledge items and in what order to show the user when she invokes LAVA.

Conversation History holds the support conversation data while Conversation Context is there to save the general conversation data like status and user context when activating the conversation.

3.1. Actors and Processes

LAVA model is based on a communication process between three actors – the User, the Service desk employee (Employee) and LAVA. The process can be described using CBR paradigm (Table 2).

Table 2. LAVA communication process (Rats and Pede, 2020).

Process	Description	Actor
RETRIEVE	Retrieves and presents to the user context related Knowledge items. Collaborative filtering (FAQ lists - items preferred by other users) and content-based approach (full-text search) used.	LAVA
REUSE	Uses items presented by LAVA, or navigates through topic hierarchy, or searches or opens a support request	User
	Logs user actions in an Activity Log and calculates the Range Matrix	LAVA
REVISE	Analyses the user support request and creates or modifies knowledge item(s)	Support
RETAIN	Saves the Knowledge item updates	LAVA

The model provides following options for a User:

- context dependent ranged FAQ list (section 3.2);
- full-text search in the Knowledge store (section 3.3);
- navigation through a hierarchy of knowledge topics (section 3.3);
- communication with Service desk staff (section 3.5).

LAVA is designed to reduce traffic of User's service requests. The reason why the user creates a service request is because she has a problem she cannot solve easily on her own. The first three options above are aimed at helping the user to find the solution thus reducing the number of service requests.

The Service desk has the following means:

- tools for handling the Knowledge store, including item filtering and search integrated in knowledge item editing page; this allows to search for similar items in Knowledge store (and to avoid creating duplicate items) and to create hyperlinks easily;
- tools for maintaining the topic hierarchy and linking knowledge items to topics
- workflow management means to handle item creation and confirmation;
- tools to handle user service requests.

3.2. Ranged context-sensitive FAQ

Table 3. Creating the ranged context-sensitive FAQ.

Log user activities	LAVA logs user activities of types - open (user opens the knowledge item), admit (user rates an item as being useful), reject (not useful) and use link (user clicks a link in the knowledge item content). Every type of activity has a weight used for range calculation. Activities are logged against user context – a UI item id of the supported product the user has activated LAVA with.
Calculate item ranges	LAVA calculates periodically the range matrix. The range matrix is calculated against the available contexts and knowledge items using the formula 1 (see below).
Assemble and present a ranged FAQ	LAVA gets the context id upon activation and retrieves the range matrix part for this id. The ranges of the knowledge items are updated then adding a constant (configurable) value to ranges of all items declared by the service desk relevant for the given context. The list of top ranges is assembled after that. Another means of knowledge item selection is used if there are not enough (a configurable amount) items in FAQ after the first step (this may occur in particular during the initial period of LAVA use in production). Each context can be associated by the service desk with a set of keywords. The keyword search is used against this keyword set to retrieve additional items for the FAQ.

One of the LAVA model priorities is to present a user with useful information as soon as she activates LAVA. This is done by the process outlined in Table 3.

The range A_{ij} for knowledge item Z_j and context S_i is calculated by the following formula (Rats and Pede, 2020).

$$A_{ij} = \begin{cases} 10, & if\ the\ Employee\ links\ Z_j to\ S_i \\ A_{ij} + \alpha, & if\ User\ opens\ Z_j for\ S_i \\ A_{ij} + \beta, & if\ User\ uses\ rates\ positively\ Z_j\ for\ S_i \\ A_{ij} + \gamma, & if\ User\ uses\ link\ Z_j\ for\ S_i \\ max(1, A_{ij} - \delta), & if\ User\ rates\ negatively\ Z_j\ for\ S_i \end{cases} \quad (1)$$

Parameters α, β, γ and δ are called *action weights*. They are integers in range [1, 9] and will be obtained from empirical evidence (see section 5.2).

We assume the same user will access a particular knowledge item merely a couple of times. There is no ground, thus, to suppose the importance of user actions should depend on time (e.g. – newer actions are more important than older ones).

3.3. Search

LAVA model exploits advanced search means of the Elasticsearch engine (WEB b; Rats, 2018):

- ranged full-text search; ranged search allows to score search results against the relevance function and retrieve top results;
- filtering against a number of knowledge context parameters (Product module, Product version range etc.).

To create ranged context-sensitive FAQ lists LAVA uses filtering and the range matrix.

LAVA model incorporates a number of Elasticsearch means to ensure as much of the relevant information is returned as possible. The custom-built text analyzer lava_latvian is created for Elasticsearch full-text search that addresses some important general and Latvian language specific issues. The steps of the lava_latvian analyzer are described in Table 4. The analyzer is used both when indexing and searching. Thus, the index of a text is lowercase with stop words excluded, converted to stemmed form and to the main form of the synonym list (if in the synonym dictionary). The same happens when searching – this ensures that when a word in some form (or one of the synonyms) searched, words in other forms (and other synonyms) are matched.

Table 4. The analyzer lava_latvian.

Step	Description
lowercase	Converts the text to lowercase.
latvian_stop	Removes from the analyzed text the words from the Latvian stop list (custom created for LAVA).
latvian_stemmer	Converts the words to the stemmed form.
lava_synonym	Converts the words from the synonym list to the main form.

It should be stressed that the lava_latvian synonym dictionary is built on stems, not words.

3.4. Topics

The topic hierarchy of the LAVA model is based on the following principles:

- There is exactly one root topic - a topic of the highest level;
- A topic is allowed to have several child topics and several parent topics;
- A topic of any level is allowed to have related knowledge items;
- The topic hierarchy should be organized in a way that a parent topic would have 3 to 8 child topics;
- Topics and knowledge items should be organized in a way to reduce as much as possible the summary distance of the knowledge items from the top.

The navigation in the topic hierarchy is as follows. The user initially is presented with the topic of the highest level. LAVA presents the user with a list of links to the next level topics and a list of links to the knowledge items related to the topic (if any). When the user clicks on a child topic she is presented with the same data for the child topic.

3.5. Communicating with Service Desk

Two-level model is frequently exploited to organize user support. Power users (the first level user service) handle routine, recurring user requests and escalate to the second level only what they are not able to address themselves. There are several ground reasons for selecting this model:

- the two-level model allows to reduce the volume of service requests at a second level that might be responsible for supporting a large number of users;
- the two-level support helps to keep inside the organization the content of the support calls that frequently contain confidential data.

Unfortunately, this model locks inside the organization as well the knowledge of potential value for other organizations. In particular, the second level service desk has little knowledge about the routine problems of the end users.

The LAVA model addresses this with its bidirectional process of the knowledge sharing. LAVA Knowledge Store is persisted in a cluster of nodes – main node (main store) plus node for each Product instance (instance store). Knowledge on instance nodes may be propagated to the main node while knowledge on the main node may be disseminated to the instance nodes. The first level support staff has control over what knowledge is propagated thus allowing to propagate useful knowledge and to protect the sensitive data at the same time.

The knowledge cluster is maintained as follows:

- The initial content of the Knowledge Store is created in the main node and disseminated to the instance nodes according to the instance licenses (e.g. what modules of the Product are licensed);

- Power users (first level support) may add knowledge to the instance store and allow (or disallow) the newly created items to be propagated to the main node.

Two attributes are introduced in the data model of the Knowledge store to control the knowledge propagation and dissemination process:

- Attribute *mode* defines if the knowledge item is considered as specific for his organization (mode S) or as generally useful (mode P); only mode P items are allowed to be propagated; the attribute is controlled by the first level support;
- Attribute *span* allows to tell apart the general knowledge items (created on the main node, span D) from the instance specific knowledge items (created on instance node, span I). The second level support is in charge of transforming I items to D items (this may or may not include some editing of the item content).

The LAVA model proposes two channels to communicate service requests:

- User comments submitted while viewing the knowledge item; those are converted by LAVA to knowledge item specific service requests;
- General service requests, submitted in a dedicated web page.

Service requests are handled by the same LAVA model process no matter what channel was used to create them.

4. Considerations for success

The success of the LAVA model is determined by two parties involved – the end users and the service desk staff. We believe the end users will use LAVA actively if it allows them to easily find quality knowledge useful to solve their current problems. Table 5 brings some details of what we mean by “easily find”, “quality knowledge” and “useful”.

It is important as well that a user has an option to communicate easily with a support staff in cases when she fails to find the answer in LAVA knowledge base. The LAVA model supports that by allowing to create as needed both general and knowledge item dedicated service requests.

It is of crucial importance to create an initial Knowledge store of such volume and quality that a User could easily find answers to the significant part of routine questions. LAVA may be deployed for the production use only after this is done as otherwise the bad first experience may discourage users from searching the solutions in the LAVA Knowledge store and revert them to addressing all questions to the service desk.

Table 5. Quality answer.

Easy to find	Ranged context-dependent FAQs, full-text search and topic navigation are the LAVA means that should help user to get the necessary knowledge quickly and easily. If user searches for solution to a routine problem the FAQ may very well contain the necessary solution. If no – the LAVA customized full-text search or navigating well-structured topics may help.
Quality knowledge	Users today are not tempted to read through lengthy manuals to find a bit of knowledge they need. The knowledge items should be structured in a way that user would search knowledge items not a knowledge inside a particular item. This means that any lengthy description of a complicated process must be split into several knowledge items. The main item should comprise a high-level description and refer to other items describing any details. The level of granularity of the Knowledge store should allow user to comprehend easily any particular knowledge bit. The granularity is important as well because knowledge items are the bits of knowledge the rating and ranging mechanism of LAVA is based on. The content and form of each knowledge item is of great value as well. Clear and easily understandable language, text formatting that emphasizes the most important notions, images and videos created to help understanding are crucial.
Useful knowledge	We assume that user activates LAVA assistant because she has a problem while using a Product to carry out a specific business task. The problem might be about what features of the Product should be used still the business problem is primary while the features of the Product are here to support business tasks. This means that useful knowledge items should be aimed at solving business tasks, not at describing the Products features.

5. Measuring the success

LAVA is meant to provide Users with a content relevant to their current needs. LAVA processes of creating FAQs and searching relevant items are parametrized hence it is important to have means for performance measuring and parameter evaluation. Below in section 5.2 we describe the parameter evaluation for FAQs creation.

5.1. LAVA model parameters

LAVA model has a number of configurable parameters:

- maximum and minimum number of items to show on FAQ list;
- minimum item range to show on FAQ list;
- action weights (see section 3.2);
- time period between consecutive range matrix creations;

- boost factor of question field (defines its relative importance for search compared to the answer field);
- the synonym dictionary etc.

The performance of the LAVA model depends on the values of the configuration parameters. E.g. the content and item order of a FAQ list depends (apart from the user actions) on action weights, minimum item range, and maximum and minimum number of items. The content and order of items returned by search, in turn, depends on the boost factor and the contents of the synonym dictionary. LAVA model uses simulation to evaluate what parameters are better. The simulation uses user action data collected by LAVA model.

5.2. Parameter evaluation for FAQs creation

The FAQ lists are meant to contain the most relevant knowledge items ordered descending by relevance range. Although each user has specific needs we should statistically observe that users open and positively evaluate items at the top more than items at the bottom of a FAQ list (or not in the list). This way it is reasonable to evaluate a parameter set from this perspective – how knowledge item positions on FAQ lists match what users select and how they evaluate. As long as positions depend on ranges and they in turn depend on the configuration parameters this gives us a method to evaluate LAVA parameters.

We base our model of parameter evaluation on assumption that positions closer to the top of a FAQ are more important and this importance can be declared to be proportional to number of expected clicks on a position. We use a google search click through analysis data to deduce an empiric position importance formula. The data in (Wadsack, 2015) shows the top position accounts for 17% to 58% of all clicks depending on a search domain. This makes a 27% on average for the top position. Other sources present similar results.

Analysing the click through data of the top 10 positions in the referred and other sources we concluded that a good approximation of the click through T_i for position i can be obtained by formula 2.

$$T_i = \frac{1}{\mu * i} \tag{2}$$

Here T_i is the probability that a user clicks on the row i. The probability p_{in} user clicks on any of the m rows is a sum of T_i for all the rows. The probability p_{in} can be calculated in our case because we have log data on what searches user executed and what rows of the search results user selected. We can execute for the given set of parameters the searches and determine if a particular item is on the search results. This way we can deduce the formula 3 to calculate parameter μ.

$$\mu = \frac{1}{p_{in}} \sum_{i=1}^{m} \frac{1}{i} \tag{3}$$

As far as we have position weights and parameter μ we can use formula 4 to calculate ranges for action types.

$$C_x = \frac{100}{\mu * T} \sum_{i=1}^{m} \frac{S_i}{i} \tag{4}$$

Where x is an action type (O - open, Y – rate positively, L – hyperlink click or N – rate negatively), T is a total number of user actions, m is a number of rows in a FAQ list, S_i – a number of times users have performed action of type x with a knowledge item in position i of the FAQ.

The overall range of the parameter configuration C is calculated then as follows (formula 5):

$$C = C_o + C_y + C_l - 3 * C_n \tag{5}$$

Where C_o, C_y, C_l and C_n are ranges for action types O, Y, L and N respectively.

Grid search (grid sampling) (Brownlee, 2021) will be used to generate parameter configurations for evaluation. The grid will be based on parameter values close to currently used.

Some other performance metrics for LAVA model are presented in (Rats and Pede, 2020).

6. Conclusions and future work

The model of a smart user Support Assistant LAVA is developed. LAVA provides a Product User with context-related knowledge that should help her to solve by herself most of the routine issues and thus reduce the amount of support requests. LAVA Knowledge Store is persisted on a cluster of nodes – main node plus instance nodes. Knowledge may be propagated from instance to the main node and disseminated from the main node to the instance nodes that enables users of the instance to keep sensitive data inside while allowing to share useful knowledge with others.

LAVA end user has several information retrieval options – context-dependent FAQ lists, customized full-text search and a hierarchy of topics. Service desk staff in turn has a number of means allowing to monitor and improve the Knowledge store (i.e. using the knowledge obtained during the service request handling) and to use its knowledge while handling service requests.

The prototype of LAVA model is developed as a web application using python with flask framework, javascript and bootstrap. Elasticsearch is used for data persistence and Kibana for visualization.

Currently the LAVA prototype is proposed for beta testing to several customers. The data collected during the testing will be used in particular for performance measurement and for evaluation and configuration of the LAVA parameters.

List of abbreviations

CBR	Case-based Reasoning
FAQ	Frequently Asked Questions
LAVA	Smart assistant for the user support (Latvian acronym).
UI	User Interface

Acknowledgements

The research has received funding from the project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/18/A/003.

References

Aamodt, A., Plaza, E. (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*. AI Communications. IOS Press.

Brownlee, J. (2021) *Random Search and Grid Search for Function Optimization*, https://machinelearningmastery.com/random-search-and-grid-search-for-function-optimization/

Fontanella, C. (2020) *The Top 14 Free Help Desk Software and Ticketing Systems in 2020*, https://blog.hubspot.com/service/free-help-desk-software

Graw, M. (2021) *Document360 review | TechRadar*, https://www.techradar.com/reviews/document360-review

Raman, V. (2017) *Recommender Engine — Under The Hood - Towards Data Science*, https://towardsdatascience.com/recommender-engine-under-the-hood-7869d5eab072

Rats, J. (2018) 'Optimizing the enterprise search', in *Proceedings - 2017 4th International Conference on Mathematics and Computers in Sciences and in Industry, MCSI 2017*. doi: 10.1109/MCSI.2017.20.

Rats, J., Pede, I. (2020) 'Using a Context Based Knowledge for Software Product User Support', in *2020 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University, ITMS 2020 - Proceedings*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ITMS51158.2020.9259307.

Sutton, R. (2020) *The Bitter Lesson of Machine Learning*, https://www.kdnuggets.com/2020/07/bitter-lesson-machine-learning.html

Trujillo, E. (a). *GitHub - mayanksingh2298/food-recommender*, https://www.groovehq.com/blog/automated-customer-service.

Wadsack, J. (2015) *Are Averaged Google Organic Search Click Through Rates Useful? - Keylime Toolbox*, https://www.keylimetoolbox.com/seo/are-averaged-google-organic-search-click-through-rates-useful/

WEB (a). *Customer Service Software for Small to Enterprise Businesses*, https://www.hubspot.com/products/service

WEB (b). *Elasticsearch: The Definitive Guide [master] | Elastic*, https://www.elastic.co/guide/en/elasticsearch/guide/master/index.html

WEB (c). *Fully featured knowledabse software – KBPublisher*, https://www.kbpublisher.com/features/

WEB (d). *Jira Service Desk | Atlassian*, https://www.atlassian.com/software/jira/service-desk/features

WEB (e). *osTicket*, https://osticket.com/features/
WEB (f). *Free Helpdesk Software | IT Helpdesk | HR Helpdesk | Admin Helpdesk*, http://www.cdesk.in/Support_Management/Support_Management.aspx
WEB (g). *Help Desk Software (IT Support Software) - Get Free Trial | SysAid*, https://www.sysaid.com/help-desk-software
WEB (h). *Help Desk Software HESK - a free PHP help desk*, https://www.hesk.com/
WEB (i). *HelpDeskZ :: Support Ticket System*, https://www.helpdeskz.com/
WEB (j). *HelpSite: Make a Knowledge Base / Support Center - for free!* https://helpsite.com/
WEB (k). *ngDesk | The best platform for your business*, https://ngdesk.com/
WEB (l). *IT help desk software key features | Features of a good help desk ticket management system*, https://www.manageengine.com/products/service-desk/help-desk-features.html
WEB (m). *Just Enough Help Desk Software | HelpSpot*, https://www.helpspot.com/help-desk-software
WEB (n). *Online FAQ Software | Build Your Free FAQ System*, https://www.proprofs.com/knowledgebase/faq-software/
WEB (o). *Support Hero*, https://www.supporthero.com/
WEB (p). *Web Help Desk - IT Ticketing Software | SolarWinds*, https://www.solarwinds.com/web-help-desk
WEB (q). *Zoho Desk | Customer Service Software for Context-Aware Support*, https://www.zoho.com/desk/

Received March 17, 2021, revised April 28, 2021, accepted May 6, 2021