FCA-based Approach for Interactive Query Refinement with IR-chatbots

Tatiana P. Makhalova[a], Dmitry A. Ilvovsky[b], Boris A. Galitsky[c] and Elizaveta F. Goncharova[b]

[a]*Universite de Lorraine, CNRS, Inria, LORIA, Nancy, France*
[b]*National Research University Higher School of Economics, Moscow, Russia*
[c]*Oracle Corp., Redwood Shores, CA, USA*

Abstract
Information retrieval (IR) chatbot is a special class of virtual assistants, which is widely used nowadays in customer support services. However, the work of modern IR retrieval systems is limited by simple queries to the database, which does not utilize all the potential of interaction with the user. In this paper we implement an FCA-based approach to deliver the relevant information the user has requested. A developing approach integrates a concept-based model build upon the database and intelligent traversal through it. The proposed algorithm has been implemented as an additional function within the existing IR chatbot. In this paper we also enlighten the perspectives for further development of the proposed system. Formal Concept Analysis (FCA) technique and Pattern Structures as its extension are proposed to process unstructured data (objects with a text description), which has become a common way of presenting various items recently.

1. Introduction

This paper considers the main principles which underlie information retrieval (IR)-chatbots [1]. This type of chatbots is varied from the “Social chatbots”, mostly oriented on the entertainment and dialogue which sounds like natural conversation [2], or “Task-oriented chatbots”, which have a goal to retrieve information in a particular domain [3], [4], [5], [6]. The IR-chatbots belong to a specific class of task-oriented chatbots (those who support web search in case of imprecise queries in specific domains). There is a list of essential features that IR-chatbots should possess.

— The main goal of this type of chatbot is to help a user to find a particular object in the database. The important thing is that the user may have only general information about the desired object.

— In most cases at the beginning of the search a user is not familiar with all the possible characteristics of the object he wants to find. However, as he knows more peculiarities

Russian Advances in Artificial Intelligence: selected contributions to the Russian Conference on Artificial intelligence (RCAI 2020), October 10-16, 2020, Moscow, Russia
tpmakhalova@hse.ru (T.P. Makhalova); dilvovsky@hse.ru (D.A. Ilvovsky); boris.galitsky@oracle.com (B.A. Galitsky); egoncharova@hse.ru (E.F. Goncharova)

of the objects belonging to a category he is interested in, he may want to refine his initial query and add some additional information.

— A chatbot has access to the database, containing information about all available objects, and metadata (hierarchically organized categories of objects).

— A chatbot should adjust to the user request keeping the variability of objects in the datasets.

— One of the main properties of the IR-chatbots is efficiency; it should propose a satisfactory result in a few iterations of communication with the user.

It should be mentioned that in most cases traditional IR-chatbots send queries to the database ignoring any knowledge models that could be built upon the database. The motivation of the proposed approach is that instead of simple queries to the database, we build a knowledge model and traverse it with the proposed two-level interactive algorithm. Interactive part allows the chatbot to adjusts to the user's preferences automatically. In this paper we aim to explain the main principles the proposed algorithms are based on and to present the experimental results obtained for real data. We also discuss the future direction of this research.

The paper is organized as follows. In Sect. 2 we present the basic notions of FCA and PS. Sect. 3 starts with a small example where IR-chatbots might be used, and then we introduce a knowledge model that is the basis for IR-chatbots. In Sect. 4 we discuss principles for the IR-chatbot functioning. In Sect. 5 we present the experimental results. We conclude and give the direction for future work in Sect. 6 and 7.

2. Basic Notions

The main theory that lies upon the proposed methods is Formal Concept Analysis (FCA) and Pattern Structures (PS). Let us introduce some basic notions of the theory.

2.1. Formal Concept Analysis

In Formal Concept Analysis theory [7] a formal context is a triple (G, M, I), where $G = \{g_1, g_2, ..., g_n\}$ is a set of objects, $M = \{m_1, m_2, ..., m_k\}$ is a set of attributes, and $I \subseteq G \times M$ is an incidence relation, i.e. if object g has the attribute $m : (g, m) \in I$. The derivation operators $(.)'$ are defined as $A' = \{m \in M | \; \forall g \in A : gIm\}, A' = \{g \in G | \; \forall m \in B : gIm\}$. Double application of derivation operator is called a closure operator $(.)''$. Sets $A \in G, B \in M$ such that $A = A''$ and $B = B''$ are said to be closed. A (formal) concept is a pair (A, B), where $A \subseteq G, A \subseteq M$ and $A' = B, B' = A$. A is called the (formal) extent, and B is called the (formal) intent of the concept (A, B). A partial order $\leq$ is defined on the set of concepts as follows: $(A, B) \leq (C, D)$ iff $A \subseteq C (D \subseteq B)$, a pair (A, B) is a subconcept of (C, D), while (C, D) a superconcept of (A, B).

2.2. Pattern Structure

As the standard definition of FCA is related to data presented in binary format, which does not always suit the real data, the Pattern Structures were introduced [8]. Pattern structures generalize formal contexts and allow operating with objects described by more complex types of attributes.

A Pattern Structure is a triple $(G, (D, \sqcap), \delta)$, where G is a set of objects, D is a set of all possible object descriptions, and $(D, \sqcap)$ is a meet-semilattice of object descriptions. Mapping $\delta : G \longrightarrow D$ takes an object g to its description $d \in (D, \sqcap)$. Galois connection between $\left(2^G, \subseteq\right)$ and $(D, \sqsubseteq)$ is defined as $A^\square = \sqcap_{g \in A} \delta(g)$, $A \subseteq G$, $d^\square = \{g \in G | d \sqsubseteq \delta(g)\}$ for $d \in (D, \sqcap)$, where $d \sqsubseteq \delta(g) \iff d \sqcap \delta(g) = d$. A pair (A, d) for which $A^\square = d$ and $d^\square = A$ is called a pattern concept.

2.3. Stability Measure for Concepts

Stability indices for formal concepts were introduced in [9], [10] and modified in [11]. Intentional stability of concept (A,B) is the probability that B will remain closed when removing a subset of objects from extent A with equal probability

$$stab((A, B)) : \frac{|\left\{C \subseteq A \mid C' = B\right\}|}{2^{|A|}}.$$

The concepts with the high values of $stab((A, B))$ are more stable regarding random removal of the objects. The computing stability is $\#P$-complete. In practice, one uses its approximations [12], [2], [13]. One of the most popular approximations is Δ-measure [14]. The Δ-measure is the minimal difference in supports between concept (A, B) and its nearest subconcepts: $\Delta((A, B)) = min_{(A^*, B^*) \leq (A, B)} |A| - |A^*|$.

3. FCA-based Approach for Query Refinement

In the beginning, we would like to consider a small use case where the described approach of building knowledge domain and its traversal can be implemented. Table 1 presents the database that is used in this example. This database is a small fragment of the database used in experiments (Section 5) with a shortened number of objects and attributes describing them.

Table 1
A fragment of the database

	Brand	Battery	Weight	Camera	GPU	RAM	Screen Size	Color	Price
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
g_1	0	9720	0.63	12		4	12.9	Silver, Gray, Pink, Gold	812
g_2	1	5124	0.3	8		3	7.9	Silver, Gold	328
g_3	1	8134	0.45	8		3	10.5	Silver, Gray	421
g_4	2	3900	2.78		G1	16	15.6	Gray	779
g_5	2	3145	1.63		G1	16	17.3	Gray	829
g_6	6	6500	1.1		G2	12	14	Silver	639

The objects $\{g_1, g_2, g_3\}$ are tablets, they belong to the same category *"Tablet"*, the objects $\{g_4, g_5\}$ are laptops, they belong to the category *"Laptops"*, while g_5 also belongs to *"Transformer-laptop"* category, g_6 is a laptop and also an ultrabook, so, it also belongs to two categories.

The scenario of IR-chatbot performance is the following, the user aims to find some items, and the chatbot provides some features that the user may refine to filter the items. As has been mentioned in the introduction, the user may know some attributes he expects to get within the item at the beginning of the search, or he may not. In the latter, the only information he possesses is that he wants to purchase some object that satisfies his requirements, which are not formulated yet.

The standard IR engine usually proposes a set of attributes that the user should refine. The attributes can be chosen in different ways, we have distinguished some of them:

— show only the most general attributes for refinement, i.e., price, brand, average customer review, etc. (such kind of refinement might be irrelevant for users);

— show the most frequently refined attributes (it does not take into account how diverse the range of goods within a specific category is and is not adapted to specific preferences of a user);

— show all attributes (this approach is rarely used in practice since it offers a wide range of options to specify and worsens the user experience).

After the features are revealed the user filters some of them, and the chatbot sends the query to the database. In the end, the user gets a huge list of objects with the set of all possible characteristics, and he/she should be able to find one (or several items) that meets all his/her requirements. This approach has an obvious drawback, a web search engine may propose irrelevant attributes to refine or does not take into account the user preference adjusted depending on the availability of goods.

The example above demonstrates the main problem of the existing tools: in case of very general queries the search engine does not provide user-friendly interface for attribute refinement, i.e., it does not provide convenient tools to discover a variety of objects the user might be interested in.

3.1. Domain Knowledge Model

As we have mentioned above this work is a logical continuation of the work [1], where we introduced the possible theoretical framework that can be implied in IR-chatbot architecture. This paper focuses on the implementation part and presentation of the experimental results. However, we would also like to give a full description of the algorithms we use, and the modification that was included in the approach in comparison to the first theoretical version of the framework.

It should be mentioned that IR-chatbots usually work with huge databases containing extremely heterogeneous items. These objects (goods) are described by completely different attributes. Even in our running example, the goods from category *"Tablet"* and *"Laptop"* have different nonoverlapping features (*"Camera"* for tablets, and *"GPU"* for laptops). If we consider more complex databases, then we can see various goods that have nothing in common. Thus, it is reasonable to use this information in the process of building a knowledge model.

A two-level knowledge model suits this problem. On the upper level the objects are divided into several very general categories, which do not overlap and using FCA theory, we build the upper-level model where the set of objects G contains the initial objects from the dataset and the attributes are binary, defining whether the object possesses the following characteristic, or not. The example of such kind of general categories is electronic devices furniture, clothes, building materials, etc. Thus, on the upper level of knowledge model, the groups of the most similar objects (in sense of having a specific characteristic) are revealed. Then, on the bottom level, we process the objects belonging to the same group (concept) more accurately, considering the specific value of each attribute.

3.2. Upper-Level Model

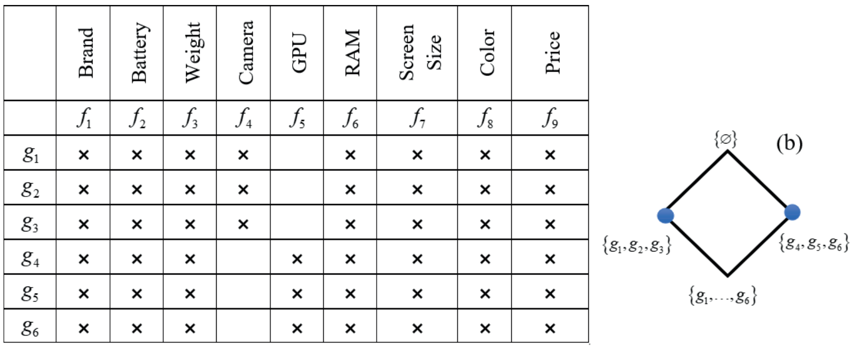
The heterogeneous objects are the ones described by different attributes that are never used together. At the upper-level model they are put into different categories, e.g., consider data from 1, the tablets have the unique attribute *"Camera"*, which other goods do not possess, so the category "Tablet" is general enough for the upper level. At the same time, all notebooks have an attribute *"GPU"*, which separates them from the objects belonging to other categories; thereafter, *"Laptop"* could be the upper-level category for these goods as well.

Let G be a set of objects, described by a set of attributes M, where every $m \in M$ has a particular domain of values $dom(m)$. We denote the set of all possible values by W, i.e., $W = dom(m) \mid m \in M$. Every object is described by a small subset of attributes from M. For objects G and attributes M we define a binary relation I as follows, $(gIm) = 1 \iff m$ is defined for g. Thus, $g' \subseteq M$ is a subset of attributes that is used to describe object g, and $A' \subseteq M$ is a subset of attributes that are used to describe every object in A. The "*is defined*" relation for objects from Table 1 is given in Figure 1 (a). The degree of objects homogeneity $A \subseteq G$, $h(A) = |g' \mid g \in A|/|M|$, is the rate of their common attributes. A sublattice of concepts with the homogeneity rate $h(A) \in [h_1, h_2]$ represents the coarse categorization model. The lower bound h_1 prevents from creating too general categories, while the upper bound h_2 allows not considering very homogeneous objects at the upper level. For example, if we get a concept with the intent which contains only one feature out of 20 initial, then its homogeneity rate $h(A)$ is 0.05, this means that this group of objects is too general. So, it is useless to launch the query refinement procedure for this group of objects, as they are extremely varied.

The upper level of the knowledge model is a fragment of a concept lattice $\mathcal{M}_{\mathcal{U}}$, computed on the "*is defined*" relation where the concepts meet the homogeneity rate requirement and augmented with the set of category names. Figure 1 shows the table represented "*is defined*" relation for the objects from Table 1 (a), and $\mathcal{M}_{\mathcal{U}}$ for that data (b).

3.3. Bottom-Level Model

Homogeneous objects, which are the groups of objects $A_i \subseteq A$, where $(A, B) \in \mathcal{M}_{\mathcal{U}}$, are described by the high rate of similar attributes. To compute refined groups $A_i \subseteq A$ we took the attributes that defined for all the objects in A and build a pattern structure on the corresponding



	Brand	Battery	Weight	Camera	GPU	RAM	Screen Size	Color	Price
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
g_1	×	×	×	×		×	×	×	×
g_2	×	×	×	×		×	×	×	×
g_3	×	×	×	×		×	×	×	×
g_4	×	×	×		×	×	×	×	×
g_5	×	×	×		×	×	×	×	×
g_6	×	×	×		×	×	×	×	×

{∅}
(b)
{g1, g2, g3}
{g4, g5, g6}
{g1, …, g6}

Figure 1: The binary relation "*is defined*" for a dataset given in Table 1

data fragment. The derivation operator $(.)^{\square}$ and intersection operator $\sqcap$ are defined differently for different attribute types and may also depend on the semantic behind an attribute. For example, the real-valued features r_1 and r_2 can be presented as the ranges $[r_1, r_1]$ and $[r_2, r_2]$; the intersection operator $\sqcap$ is defined as follows, $r_1 \sqcap r_2 = [r_1, r_1] \sqcap [r_2, r_2] = [min(r_1, r_2), max(r_1, r_2)]$. Thus, we get refined groups of objects within each pattern structure and denote these groups by $\mathcal{M}_{\mathcal{B}}(A)$. This pattern structure is the basis for further interaction with the user and the refinement of his query.

4. Interactive Query Refinement

4.1. Basic Idea of Two-stage Approach

The basic idea of query refinement was introduced in [1]. We revise the main points of the proposed algorithms and include slight modifications that were revealed during the implementation process.

The two-level knowledge model described above is used by chatbots to improve user search experience in case of imprecise queries. The proposed approach supposes the interactive manner of query refinement during the traversing of the bottom level model. As the result, the chatbot working by the proposed scheme is flexible in terms of considering both user preferences and the available set of objects with consequently specified features.

The interactive query refinement consists of two main steps:

1. Navigating to the group of homogeneous objects, i.e. a formal concept $(A, B) \in \mathcal{M}_{\mathcal{U}}$ at the upper level.
2. Query refinement within a group of homogeneous objects A by walking through pattern concepts in $\mathcal{M}_{\mathcal{B}}(A)$ at the bottom level.

4.2. Query Refinement

Let us consider the second stage of the proposed algorithm, because several slight modifications have been included there in comparison to the previous version [1] of theoretical framework.

The pseudocode for query refinement on the bottom level of knowledge model is given in Algorithm 1.

The set of homogeneous objects and the feature that describe it are directed into the bottom level of knowledge model in the form of formal concept $(A, B) \in \mathcal{M}_{\mathcal{V}}$. The algorithm working at this level calculates the corresponding pattern structure $\mathcal{M}_{\mathcal{B}}(A)$ and continues the process of query elaboration.

The algorithm starts working with a pattern structure $\mathcal{M}_{\mathcal{B}}(A)$ and, optionally, values V of some attributes from M. The basic idea of the algorithm is to walk through pattern structure jumping from one promising concept to another, interacting with the user simultaneously and updating the user request. After an iteration the chatbot specifies the concept he proposes to the user, thereafter, the user gets the most specific concept, containing one or several elements from the database. The user might be satisfied with the result, then the goal of interaction is achieved, or not (if the remained goods contain features which are also not suitable for the user). If the final result is negative, then the chatbot can start another session and try to find suitable goods, starting from another promising concept.

The chatbot starts its work from the most general concept or the most specific concept that satisfies user initial request (considering values V of some attributes from M), i.e. (A_s, d_s). The most promising concepts are the lower neighbors of (A_s, d_s). These neighbors are ranked w.r.t. Δ-measure and the number of lower neighbors. Both of these characteristics are referred to the concept stability. Then starting with the most "stable", in accordance with Δ-measure, concept the algorithm uses *isVaried* function to reveal the attributes that have the most diverse value inside the concept. Then these attributes are offered to the user for refinement.

Thus, the chatbot does not propose to the user the whole range of values $dom(\cdot)$. This specification facilitates the decision-making process. The specified values W_{spec} are used to find the most general concept (A_s, d_s) where all objects have values W_{spec}. Its lower neighbors are possible groups of more specific concepts. We sort them by Δ-measure and the number of lower neighbors. Then a new iteration of the query refinement is launched.

Algorithm 1: Query refinement

```
Data: $\mathcal{M}_{\mathcal{B}}(A)$, $M$, $V$
Result: $True$, if the used have found the item(s) he searched for, $False$ otherwise
$run \longleftarrow False$;
$relevant \longleftarrow \varnothing$ // set of relevant objects;
$A_s, d_s) \longleftarrow findTheMostSpecificConcept(\mathcal{M}_{\mathcal{B}}(A), V)$;
$GS_{ranked} \longleftarrow$
  $sortByDelta(LowerNeighbors(A_s, d_s))$ // first concepts proposed for refinement;
$M_{irrel} \longleftarrow \varnothing$ // attributes marked by user as irrelevant;
while $GS_{ranked} \neq \varnothing$ or not $run$ do
    $(A_r, d_r) \longleftarrow pop(GS_{ranked})$;
    $relevant \longleftarrow A_r$;
    $M^* =$
      $M_i \mid M_i \in M \setminus M_{irrel}, M_i \in isVaried(A_r, M)$ // the set of attributes for possible refinement;

    if $M^* \neq \varnothing$ then
        $M^*_{spec} = ask(M^*)$;
        $M_{irrel} \longleftarrow M^*_{spec}$// the attributes the user does not want to refine;
        $S \longleftarrow \{b \mid b \in values(M_i), M_i \in M^*_{spec}\}$;
         // ask the user, if he is already satisfied with the results, he type εyesε, and $run \longleftarrow$
          $True$
        run, W$_{spec} \longleftarrow userChoice(S)$;
        if $W_{spec} \neq \varnothing$ then
            $(A_s, d_s) \longleftarrow findTheMostGeneralConcept(A_r, d_r), W_{spec})$;
            $GS_{ranked} \longleftarrow sortByDelta(LowerNeighbors(A_s, d_s))$;
        end
    end
end
return $run$
```

As the possible improvement for the algorithm we consider including the additional measure for ranking the concepts. In the current version of the algorithm the concepts are ranked only by Δ-measure, which defines stability. However we can also include some statistical measures about the attributes, for instance, if the most frequently refined attributes are varied inside the concept, then it may have greater weight than other concepts because in general users are interested in this specific attribute. Another variant is to apply the machine learning technique of informative feature selection into this step. For example, if we have some rate given by the users about the product we can calculate which features are the most essential for determining the high grade of the object, thereafter the concepts with varied informative features can be ranked higher in comparison to the others. Let us illustrate the principles of query refinement using the running example. Having built a patter structure for a set of homogeneous objects we got the most general concept with extent $\{g_1, g_3\}$, $\{g_2, g_3\}$, and $\{g_1, g_2\}$. The obtained concepts are similar to each other in comparison of Δ-measure and the number of lower neighbors, so the chatbot can start refinement with any concept. For the extent g_1, g_3 the most varied attributes were real-valued characteristics which are the “battery” and “screen size”. The user did not want to continue refinement because the proposed range of screen sizes did not meet his expectations, the user understood that he wanted the screen size to be smaller than 10.5. The next attempt was successful, and the user managed to find the relevant attribute after the one iteration. Figure 2 illustrates the refining process for the described case.

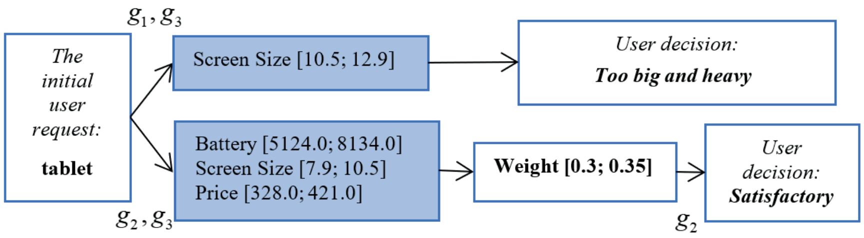


Figure 2: Dark rectangles correspond to the chatbot proposal, the attributes in bold in light rectangles are user choices

This running example does not show the advantages of using the proposed method, because the number of objects is extremely small, however, it illustrates the principles of the work.

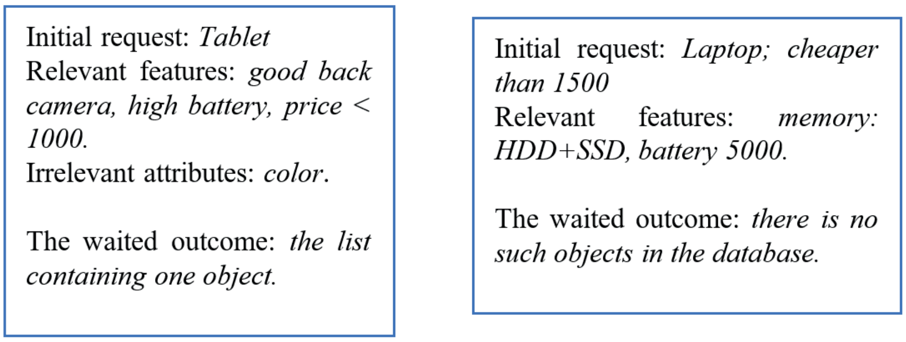
5. Experiments

The experiments were performed using the real data provided by online stores available on the Internet. The names of brands and other confidential information were replaced by some universal names, like *“Brand 1”*, *“Brand 2”*, etc.

The data is presented in the form of the dataset containing information on various types of PCs, such as laptops, ultrabooks, tablets, and smartphones, 200 objects in general. Each of them is described by various attributes, the 16 main characteristics are overviewed. However, as it already has been mentioned, the items belonging to different categories possess different attributes. Thus, the first part of the proposed methods allows us to identify the group of objects sharing similar attributes and work with them separately. The observed characteristics are the follows: *Brand*, *Battery*, *Camera resolution*, *Weight*, *Warranty*, *Ports*, *Processor*, *OS*, *RAM*, *GPU*, *CPU*, *Size*, *4G LTE*, *Sensor*, *Color*, and *Price*.

The experiments were held as follows, the users were given several scenarios of interaction with the chatbot including the initial request and the set of relevant and irrelevant attributes. The scenarios were generated randomly, the object desired by the user may be in a dataset, then the expected outcome is the list of objects that satisfies users’ request, or the user might want the object that is not contained in the database, then the final step of the interaction is the empty list, and the previous step provides the user with the objects which are the closest to the desired one.

The possible scenarios are presented in Figure 3.


Initial request: *Tablet*
Relevant features: *good back camera, high battery, price < 1000.*
Irrelevant attributes: *color.*

The waited outcome: *the list containing one object.*

Initial request: *Laptop; cheaper than 1500*
Relevant features: *memory: HDD+SSD, battery 5000.*

The waited outcome: *there is no such objects in the database.*

Figure 3: The sample of scenarios given to users during the experiments. In the first scenario the user has the requests that satisfy at least one object in the database. The second scenario does not allow the user to find the desired item, however, the chatbot may offer the closest one.

Table 2 presents the results of the experiment. During the experiments we have evaluated the number of correct outcomes when the required item was in the database (**"Satisfaction with the result *(positive)*"** column), and the number of correct outcomes when the required item was not in the initial database (**"Satisfaction with the result *(negative)*"** column). This metric measures how many times the chatbot has found the relevant items, if they are in the initial database, or has revealed that the desired item cannot be found when it is true in reality. Secondly, we calculate the average number of iterations required by the chatbot to obtain the final result for the user (**"The average number of iteration till the final result"** column). This characteristic shows how many times the user needs to refine his query in order to obtain a satisfactory result.

Table 2
The experimental results

Satisfaction with the result (positive)	Satisfaction with the result (negative)	The average number of iterations till the final result
49	10	4.7

As we saw in the previous section the number of iterations can depend on the choice of the initial concept that we choose in accordance with Δ-measure. So, we calculate the total number of iteration within all the attempts of the chatbot and normalize it by the total number of the performed scenarios, which is 60.

Thus, experimental results show that the proposed architecture works successfully. In one case only the chatbot could not find the item that meets user's requirements fully. The mistake was obtained in the scenario when the user requested the tablet with the screen size more than 13. The chatbot found 10 items with the screen size from 17 to 21 inches and stopped the interaction process, while the user wanted to specify this characteristic more and choose only the tablets, which screen size is 17. This mistake was obtained due to the specificity of *isVaried* function. The other cases were performed without any mistakes.

Some peculiarities were revealed when the chatbot processed the “negative” scenarios, where the desired item could not be found in the database. For example, if the user asked for a tablet, cheaper than 800, with the battery more than 8000, and lighter than 0.4 and the user inserted these requirements sequentially, step by step, then the chatbot in three iterations revealed that the desired item cannot be found in the database, however, it also found the list of two items with slightly larger weight (0.456, 0.483). So, in this case the chatbot also provided the list of the items which had similar characteristics to the desired one. In the case when all three limits on the features were written simultaneously, the chatbot just said that it was not able to find the requested element, and the list of the closest items was not proposed.

The average number of iterations the chatbot performs is 4.7, which is a satisfactory result.

6. Research Directions

Finally, we would like to discuss the ideas about the further direction of the research. In future work we are planning to process unstructured data or descriptions written in natural language. It is a known fact that besides some structured information, contained in the database, the objects can also be described by various data presented in the text form, which is also a great source of information. This could be the users’ feedback on the product or just a general description that the object has. The search based on this data can be helpful when the user is trying to find some items, but he also desires to know what advantages and disadvantages were revealed by the users of this item in real life.

There the proposed technique without modifications cannot be applied, because we do not have a clear structure (like features and their values) for each item. The FCA has been developed as the tool for working with the text data using Pattern structures and so called parse thickets. Using some NLP techniques we are able to create the features from the texts, so called informative parts of the texts, and then use some similarity measure to unite the items in the one concept. For instance, let us consider the phrases “this model has weak battery”, “the work without charging is so little”, “The autonomous work is frustrating”. All of these phrases represent the common feature of several models of the laptop which is a bad battery, however, all of them have several different forms, but we need to reveal their similarity This can be done by using some widely spread embedding vectors to detect whether the phrase has something in common, or not. The other problem is to detect Δ-measure for this kind of data, w.r.t., how we can measure the stability of concept with a text description. Thus, our future research will be dedicated to the generalization of the proposed model to the task of processing the unstructured data.

7. Conclusion

In this paper, we discovered a model of the IR-chatbot based on two-stage query refinement using FCA theory and Pattern Structures. We overviewed the basic principles of its work and presented the experimental results obtained with the real data.

The experiments showed that the chatbot is effective in refining the users' queries. In only one scenario out of sixty the chatbot was not able to give a satisfactory result; in other cases the goods revealed by the chatbot met all the users' requirements. It should be mentioned that the chatbot needed 4.7 iterations on average to provide a user with the final list of items.

In our work Δ-measure was used to evaluate the stability of concepts and make the process of query refinement more effective, we also mentioned some other techniques that could be helpful to estimate the importance of the concept, which are some statistical metrics, or machine learning techniques for feature selection. We will try to implement these approaches in our future works.

Overall, this novel model may compete with the traditional IR-chatbots that perform simple queries to the database. Firstly, proposed chatbot adjusts to both the user's request and the range of items that satisfy user's criterion; secondly, it helps to reveal essential features, the user might not think about in the beginning, and, finally, it simplifies the decision making process for user, because he needs to observe only small number of the offered attributes.

As the perspectives for our future investigation we revealed the new task of web search based on unstructured data or text descriptions. We believe that a similar approach can be applied not only to the structured data, when we have the "object-feature" matrix, but also to the task when each object is described by the texts using Pattern Structures and some modifications of Δ-measure.

References

[1] T. P. Makhalova, D. A. Ilvovsky, B. A. Galitsky, Information retrieval chatbots based on conceptual models, in: Graph-Based Representation and Reasoning, Springer International Publishing, Cham, 2019, pp. 230–238.

[2] K. K. Bowden, S. Oraby, A. Misra, J. Wu, S. Lukin, M. Walker, Data-Driven Dialogue Systems for Social Agents, Springer International Publishing, Cham, 2019, pp. 53–56. URL: https://doi.org/10.1007/978-3-319-92108-2_6. doi:10.1007/978-3-319-92108-2_6.

[3] M. Eric, L. Krishnan, F. Charette, C. D. Manning, Key-value retrieval networks for task-oriented dialogue, in: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL), 2017.

[4] M. Henderson, B. Thomson, J. D. Williams, The second dialog state tracking challenge, in: Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL), 2014, pp. 263–272.

[5] R. Higashinaka, K. Imamura, T. Meguro, C. Miyazaki, N. Kobayashi, H. Sugiyama, T. Hirano, T. Makino, Y. Matsuo, Towards an open-domain conversational system fully based on natural language processing, in: COLING, 2014.

[6] L. Hirschman, Evaluating spoken language interaction: experiences from the darpa spoken language program, 1998.

[7] B. Ganter, R. Wille, Formal concept analysis: Logical foundations, 1999.

[8] B. Ganter, S. O. Kuznetsov, Pattern structures and their projections, in: Conceptual Structures: Broadening the Base, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 129–142.

[9] S. O. Kuznetsov, Stability as an estimate of the degree of substantiation of hypotheses derived on the basis of operational similarity, Automatic Documentation and Mathematical Linguistics 24 (1990) 21–29.

[10] S. O. Kuznetsov, On stability of a formal concept, Annals of Mathematics and Artificial Intelligence 49 (2007) 101–115. doi:10.1007/s10472-007-9053-6.

[11] S. O. Kuznetsov, S. A. Obiedkov, C. Roth, Reducing the representation complexity of lattice-based taxonomies, 2007, pp. 241–254. doi:10.1007/978-3-540-73681-3_18.

[12] M. A. Babin, S. O. Kuznetsov, Approximating concept stability, in: Formal Concept Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 7–15.

[13] A. V. Buzmakov, S. O. Kuznetsov, A. Napoli, Scalable estimates of concept stability, in: C. V. Glodeanu, M. Kaytoue, C. Sacarea (Eds.), Formal Concept Analysis, Springer International Publishing, Cham, 2014, pp. 157–172.

[14] A. V. Buzmakov, S. O. Kuznetsov, A. Napoli, Sofia: how to make fca polynomial?, in: Proceedings of the 4th International Conference on What can FCA do for Artificial Intelligence?-Volume 1430, CEUR-WS.org, 2015, pp. 27–34.