Chapter 3
Explainable Machine Learning for Chatbots

Abstract Machine learning (ML) has been successfully applied to a wide variety of fields ranging from information retrieval, data mining, and speech recognition, to computer graphics, visualization, and human-computer interaction. However, most users often treat a machine learning model as a black box because of its incomprehensible functions and unclear working mechanism (Liu et al. 2017). Without a clear understanding of how and why a model works, the development of high performance models for chatbots typically relies on a time-consuming trial-and-error process. As a result, academic and industrial ML chatbot developers are facing challenges that demand more transparent and explainable systems for better understanding and analyzing ML models, especially their inner working mechanisms.

In this Chapter we focus on explainability. We first discuss what is explainable ML and how its features are desired by users. We then draw an example chatbot-related classification problem and show how it is solved by a transparent rule-based or ML method. After that we present a decision support-enabled chatbot that shares its explanations to back up its decisions and tackles that of a human peer. We conclude this chapter with a learning framework representing a deterministic inductive approach with complete explainability.

3.1 What Kind of Machine Learning a Chatbot Needs

3.1.1 Accuracy vs Explainability

The question of whether accuracy or explainability prevails in an industrial machine learning systems is fairly important. The best classification accuracy is typically achieved by black-box ML models such as Support Vector Machine, neural networks or random forests, or complicated ensembles of all of these. These systems are referred to as black-boxes and their drawbacks are frequently cited since their inner workings are really hard to understand. They do not usually provide a clear explanation of the reasons they made a certain decision or prediction; instead, they just

B. Galitsky, *Developing Enterprise Chatbots*,
https://doi.org/10.1007/978-3-030-04299-8_3

output a probability associated with a prediction. One of the major problem here is that these methods typically require extensive training sets.

On the other hand, ML methods whose predictions are easy to understand and interpret frequently have limited predictive capacity (inductive inference, linear regression, a single decision tree) or are inflexible and computationally cumbersome, such as explicit graphical models. These methods usually require less data to train from, but give lower classification accuracies.

Our claim in this study for industrial applications of ML is as follows. Whereas companies need to increase an overall performance for the totality of users, individual users mostly prefer explainability. Users can tolerate wrong decisions made by the companies' ML systems as long as they understand why these decisions were made. Customers understand that any system is prone to errors (Galitsky and de la Rosa 2011), and they can be positively or negatively impressed by how a company rectifies these errors. In case an error is made without an explanation, and could not be fixed reasonably well and communicated properly, customers frequently want to stop being customers of this business.

We will back up this observation, automatically analyzing customer complaints. To do that, we develop a machinery to automatically classify customer complaints with respect to whether explanation was demanded or not. This is a nontrivial problem since complaint authors do not always explicitly write about their intent to request explanation. We then compare the numbers of customers just complaining about problems associated with products and services and those requesting explanations associated with these problems. We estimate the proportion of those complaints, which require explanations.

3.1.2 Explainable vs Unexplainable Learning

To tackle the challenges associated with the lack of explainability of most popular modern ML algorithms, there are some initial efforts on interactive model analysis. These efforts have shown that interactive visualization plays a critical role in understanding and analyzing a variety of machine learning models. Recently, DARPA (2016) released Explainable Artificial Intelligence proposal to encourage research on this topic. The main goal of XAI is to create a suite of machine learning techniques that produce explainable models to enable users to understand, trust, and manage the emerging generation of AI systems (Gilpin et al. 2018).

There have been attempts to augment the learning models intrinsically lacking explainability with this feature. ML models can be trained to automatically map documents into abstract concepts such as semantic category, writing style, or sentiment, allowing categorizing a large corpus. Besides predicting the text's category, it is essential to understand how the categorization process arrived to a certain value. (Arras et al. 2017) demonstrate that such understanding can be achieved by tracing the classification decision back to individual words using layer-wise relevance propagation, a recently developed technique for explaining predictions of complex non-linear classifiers. The authors trained two word-based ML models, a CNN and a bag-of-words SVM classifier, on a topic categorization task and applied the layer-wise relevance propagation method to decompose the predictions of these models onto words. Resulting scores indicate how much individual words contribute to the overall classification decision. This enables one to distill relevant information from text documents without an explicit semantic information extraction step. The authors further used the word pair-wise relevance scores for generating novel vector-based document representations which capture semantic information. Based on these document vectors, a measure of model explanatory power was introduced and showed that, although the SVM and CNN models perform similarly in terms of classification accuracy, the latter exhibits a higher level of explainability which makes it more comprehensible for humans and potentially more useful for other applications.

Although ML models are widely used in many applications due to high accuracy, they fail to explain their decisions and actions to users. Without a clear understanding, it may be hard for users to leverage their knowledge by their learning process and achieve a better prediction accuracy. As a result, it is desirable to develop more explainable machine learning models, which have the ability to explain their rationale and convey an understanding of how they behave in the learning process. The key challenge here is to design an explanation mechanism that is tightly integrated into the ML model. Accordingly, one interesting future work is to discover which parts in an ML model structure explains its different functions and plays a major role in the performance improvement or decline at each iteration. One possibility is to better back up both the model and the decisions made. In particular, (Lake et al. 2015) proposed a probabilistic program induction algorithm, having developed a stochastic program to represent concepts, which are formed compositionally from parts and spatial relations. (Lake et al. 2015) showed that their algorithm achieved human-level performance on a one-shot classification task. However, for the tasks that have abundant training data, such as object and speech recognition, deep learning approaches still outperform (Lake et al. 2015) algorithm. There is still a long path to proceed towards more explainable deep learning decisions.

Following a recent progress in deep learning, ML scientists are recognizing the importance of understanding and interpreting what goes on inside these black box models. Recurrent neural networks have recently improved speech recognition and translation, and these powerful models would be very useful in other applications involving sequential data. However, adoption has been slow in domains such as law, finance, legal and health, where current specialists are reluctant to let an explanation-less engine make crucial decisions. (Krakovna and Doshi-Velez 2016) suggests to make the inner workings of recurrent neural networks more interpretable so that more applications can benefit from their power.

Convolutional neural networks have achieved breakthrough performance in many pattern recognition tasks such as image classification. However, the development of high-quality deep models typically relies on a substantial amount of trial-and- error, as there is still no clear understanding of when and why a deep model works. (Liu et al. 2017) presents a visual analytics approach for better understanding, diagnosing, and refining deep convolutional neural networks. The authors simulated convolutional neural networks as a directed acyclic graph. Based on this formulation, a hybrid visualization is developed to visualize the multiple facets of each neuron and the interactions between them. The authors also introduced a hierarchical rectangle-packing algorithm and a matrix re-shuffling method to show the derived features of a neuron cluster. They also proposed a bi – clustering-based edge merging algorithm to minimize visual distortion caused by a large number of connections between neurons.

3.1.3 Use Cases for the ML System Lacking Explainability

Although ML is actively deployed and used in industry, user satisfaction is still not very high in most domains. We will present three use cases where explainability and interpretability of machine learning decisions is lacking and users experience dissatisfaction with certain cases.

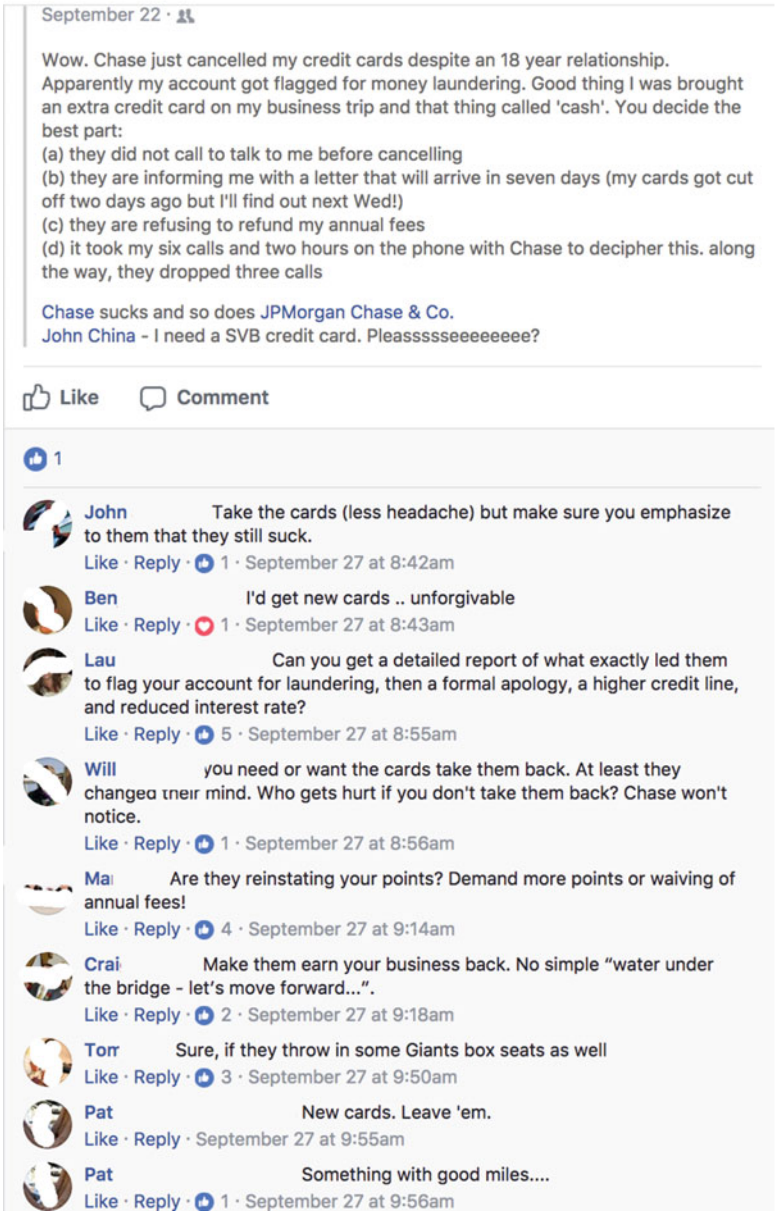
A customer of financial services is appalled when he traveled and his credit cards were canceled without an obvious reason (Fig. 3.1). The customer explains what had happened in details and his Facebook friends strongly support his case again the bank. Not only the bank made an error in its decision, according to what the friends write, but also it is unable to rectify it and communicate it properly.

If this bank used a decision-making system with explainability, there would be a given cause of its decision. Once it is established that this cause does not hold, the bank is expected to be capable of reverting its decision efficiently and retaining the customer.

An example of a popular machine learning system is shown in Fig. 3.2. The system translates the term *coil spring* (in Russian) into *spring spring*. This example shows problem in the simplest case of translation where a meaning of two words needs to be combined. A simple meta-reasoning system, a basic grammar checking component or an entity lookup would prevent this translation error under appropriate compartmental ML architecture with explainability. However, a black-box implementation of machine translation breaks even in simple cases like this. Inverse translation is obviously flawed as well (in the middle of Fig. 3.2). The bottom shows the fragment of a Wikipedia page for the entity.

Search engine is another application area for ML where relevance score is a major criterion to show certain search results (Fig. 3.3). Having a highest relevance score does not provide an explanation that the results are indeed relevant. Typical relevance score such as TF*IDF is hardly interpretable; search highlighting features are helpful but the search engine needs to be able to explain *why* it ignored certain keywords like *non-sufficient funds*. A better phrase handling would also help: the system should recognize the whole expression *non-sufficient funds fee* and if it does not occur in search results, explain it.

To investigate how important it is for a customer to have a company's decision explained, to have a decision associated with financial service *interpretable and compatible with common sense*, we need the following. A high number of scenarios of financial service failure have to be accumulated and a proportion of those requiring



September 22 ·

Wow. Chase just cancelled my credit cards despite an 18 year relationship. Apparently my account got flagged for money laundering. Good thing I was brought an extra credit card on my business trip and that thing called 'cash'. You decide the best part:
(a) they did not call to talk to me before cancelling
(b) they are informing me with a letter that will arrive in seven days (my cards got cut off two days ago but I'll find out next Wed!)
(c) they are refusing to refund my annual fees
(d) it took my six calls and two hours on the phone with Chase to decipher this. along the way, they dropped three calls

Chase sucks and so does JPMorgan Chase & Co.
John China - I need a SVB credit card. Pleasssssseeeeeeee?

John Take the cards (less headache) but make sure you emphasize to them that they still suck.
Like · Reply · 1 · September 27 at 8:42am

Ben I'd get new cards .. unforgivable
Like · Reply · 1 · September 27 at 8:43am

Lau Can you get a detailed report of what exactly led them to flag your account for laundering, then a formal apology, a higher credit line, and reduced interest rate?
Like · Reply · 5 · September 27 at 8:55am

Will you need or want the cards take them back. At least they changed their mind. Who gets hurt if you don't take them back? Chase won't notice.
Like · Reply · 1 · September 27 at 8:56am

Ma Are they reinstating your points? Demand more points or waiving of annual fees!
Like · Reply · 4 · September 27 at 9:14am

Crai Make them earn your business back. No simple "water under the bridge - let's move forward...".
Like · Reply · 2 · September 27 at 9:18am

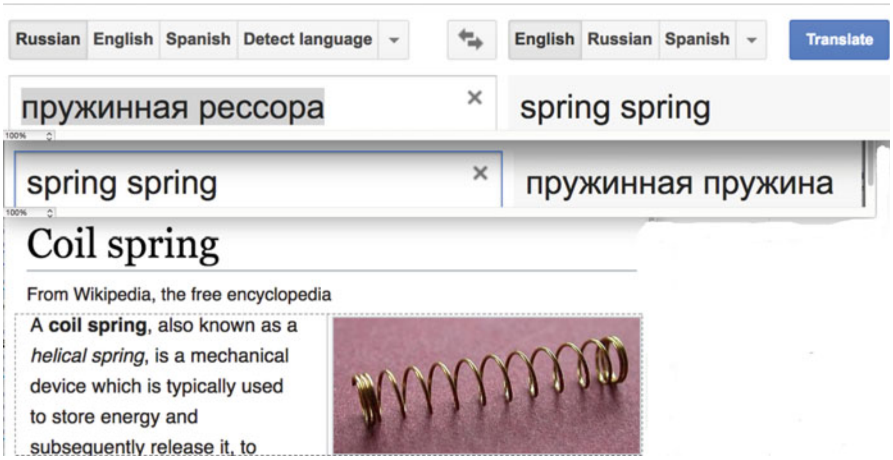
Tom Sure, if they throw in some Giants box seats as well
Like · Reply · 3 · September 27 at 9:50am

Pat New cards. Leave 'em.
Like · Reply · September 27 at 9:55am

Pat Something with good miles....
Like · Reply · 1 · September 27 at 9:56am

Fig. 3.1 A customer is confused and his peers are upset when his credit card is canceled but no explanation is provided

explanation from the company in one form or another has to be assessed. To do that, we form a dataset of customer complaint scenarios and build an automated assessment framework to detect the cases where explainability is requested.



пружинная рессора

spring spring

spring spring

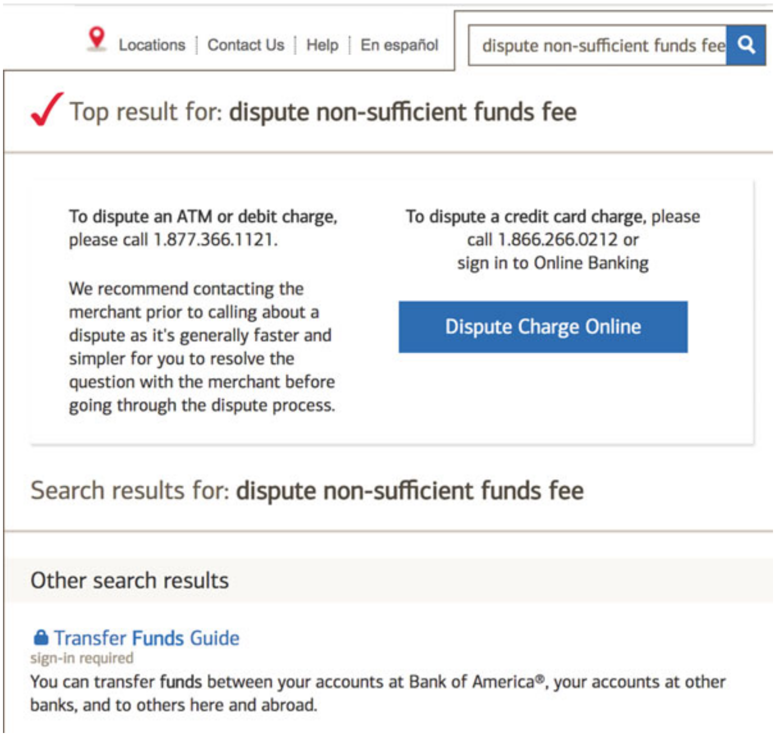
пружинная пружина

Coil spring

From Wikipedia, the free encyclopedia

A **coil spring**, also known as a *helical spring*, is a mechanical device which is typically used to store energy and subsequently release it, to

Fig. 3.2 Translation results for a simple phrase shows the problems in handling context



dispute non-sufficient funds fee

Top result for: **dispute non-sufficient funds fee**

To dispute an ATM or debit charge, please call 1.877.366.1121.

We recommend contacting the merchant prior to calling about a dispute as it's generally faster and simpler for you to resolve the question with the merchant before going through the dispute process.

To dispute a credit card charge, please call 1.866.266.0212 or sign in to Online Banking

Search results for: **dispute non-sufficient funds fee**

Other search results

Transfer Funds Guide
sign-in required
You can transfer funds between your accounts at Bank of America®, your accounts at other banks, and to others here and abroad.

Fig. 3.3 A search engine shows results very far from what a user is asking and do not attempt to explain how they were obtained

3.1.4 Automated Detection of a Request to Explain

Obviously, just relying on keywords, using keyword rules is insufficient to detect implicit request to explain. Hence an ML approach is required with the training dataset with text including a request to explain and not including one. Not just syntax level but discourse-level features are required when a request to explain is not explicitly mentioned. We select the Rhetoric Structure Theory (Rhetoric Structure Theory (RST, Mann and Thompson 1988) as a means to represent discourse features associated with affective argumentation (Galitsky et al. 2009).

Once we developed our algorithm for explanation request detection, we want to train it, test it and verify how consistent its results are across the domains. We also test how recognition accuracy varies for cases of different complexity.

Detection accuracy for explanation request for different types of evidence is shown in Table 3.1. We consider simpler cases where the detection occurs based on phrases, in the top row. Typical expressions here have an imperative form such as *please explain/clarify/motivate/comment.* Also, there are templates here such as *you did this but I expected that . . . you told me this but I received that.*

The middle row contains the data on higher evidence implicit explanation request case, where multiple fragments of DTs indicate the class. Finally, in the bottom row, we present the case of the lower confidence for a single occurrence of a DT associated with an explanation request. The second column shows the counts of complaints per case. The third column gives examples of expressions (which include keywords and phrase types) and rhetorical relations which serve as criteria for implicit explanation request. Fourth, fifth and sixth columns presents the detection rates where the complaints for a given case is mixed with a 100 of complaints without explanation request, representing the real-world situation.

Recognition accuracies, bank-specific topics of complaints and an overall proportion of the complaints with explanation request are shown in Table 3.2. We used 200 complaints for each bank to assess the recognition accuracies for explanation request. 82 ± 3% looks like a reasonable estimate for recognition accuracy for explanation request. The last column on the right shows that taking into account <20% error rate in explanation request recognition, 25 ± 4% is an adequate estimate of complaints requiring explainability in implicit or explicit form, given the set of 800 complaints. Hence the explanation request (ER) rate is about a quarter of all complaints.

Table 3.1 Cases of explanation requests and detection accuracies for model development and evaluation

Evidence	#	Criteria	P	R	F1
Imperative expression with communicative action *explain*	44	Keywords: *explain, clarify, make clear, why did they act-VP, why was it*	92	94	93.0
Double, triple+ implicit mention	67	Multiple rhetoric relation of *contrast, attribution, sequence, cause*	86	83	84.5
Single implicit mention	115	A pair of rhetoric relation chains for *contrast* and *cause*	76	80	77.9

Table 3.2 Discovering explanation request rates for four banks

Source	#	Main topics of complaints	P	R	F1	ER rate
Bank of America	200	NSF, credit card interest rate raise	82	84	83.0	**28.5**
Chase Bank	200	NSF, foreclosure, unexpected card cancellation	80	82	81.0	**25.8**
Citibank	200	Foreclosure, mortgage application, refinancing,	79	83	81.0	**23.8**
American express	200	Card application, NSF, late payment	83	82	82.5	**27.0**

Finally, we ran our explanation request detection engine against the set of 10,000 complaints scraped from PlanetFeedback.com and observed that 27% of complainants explicitly or implicitly require explainability from companies for their decisions. There is a single complaint per author. Our observation is that since almost a quarter of customers strongly demand and rely on explainability of the companies' decisions, these customers are strongly affected by the lack of explainability and may want to switch to another service. Hence the companies need to employ ML algorithms with explainability feature. A very small number of customers complained about errors in decisions irrespectively of how these errors were communicated (a manual analysis). Hence we conjecture that customers are affected by a lack of explainability in a much higher degree than by an error rate (such as extra 10%, based on anecdotal evidence) of a company's decision-making system.

This explainability feature is more important than the recognition accuracy for the customers, who understand that all businesses make errors in the course of normal operations. Typically, when a company makes a wrong decision via ML but then rectifies it efficiently, a complaint does not arise. The most important means for customer retention is then properly communicating with them both correct and possibly erroneous customer decisions (not quantitatively evaluated in this study).

Having presented the value of explainability and transparency in an ML system, we proceed to an example of a hybrid rule-based classification system. It possesses desired features and performs an essential task for chatbot functioning.

3.2 Discriminating Between a User Question and User Request

One of the essential capabilities of a chatbot is to discriminate between a request to commit a transaction and a question to obtain some information (Galitsky and Ilvovsky 2017). Usually, these forms of user activity follow each other.

Before a user wants chatbot to perform an action (such as *open a new bank account*) she would want to know the rules and conditions for this account. Once the user knowledge request is satisfied, she makes a decision and orders a transaction. Once this transaction is completed by the chatbot, the user might want to know her list of options available and asks a question (such as *how to fund this new account*). Hence user questions and transactional requests are intermittent and need to be recognized reliably.

Errors in recognizing questions vs transactional requests are severe. If a question is misinterpreted and an answer to a different question is returned, the user can reformulate it and ask again. If a transactional request is recognized as a different (wrong) transaction, the user will understand it when the chatbot issues a request to specify inappropriate parameters. Then the user would cancel her request, attempt to reformulate it and issue it again. Hence chatbot errors associated with wrongly understood questions and transactional requests can be naturally rectified. At the same time, chatbot errors recognizing questions vs transactional requests would break the whole conversation and the user would be confused on how to continue conversation. Therefore, the chatbot needs to avoid this kind of errors by any means.

Recognizing questions vs transactional requests must be *domain-independent*. In any domain a user might want to ask a question or to request a transaction, and this recognition should not depend on the subject. Whereas a chatbot might need training data from a chatbot developer to operate in a specific domain (such as personal finance), recognizing questions vs transactional requests must be a capability built in advance by a chatbot vendor, before this chatbot will be adjusted to a particular domain.

We also target recognition of questions vs transactional requests to be in a *context-independent* manner. Potentially there could be any order in which questions are asked and requests are made. A user may switch from information access to a request to do something and back to information access, although this should be discouraged. Even a human customer support agent prefers a user to first receive information, make a decision and then request an action (make an order).

A request can be formulated *explicitly or implicitly*. *Could you do this* may mean both a question about the chatbot capability as well as an implicit request to *do this*. Even a simple question *what is my account balance* may be a transactional request to select an account and execute a database query. Another way to express a request is via mentioning of a desired state instead of explicit action to achieve it. For example, utterance "I am too cold" indicates not a question but a desired state that can be achieved by turning on the heater. If no available action is associated with "cold" this utterance is classified as a question related to "coldness". To handle this ambiguity in a domain-independent manner, we differentiate between a questions and a transactional requests *linguistically*, not pragmatically.

Although a vast training dataset for each class is available, it turns out the a rule-based approach provides an adequate performance. For an utterance, classification into a request or question is done by a rule-based system on two levels:

1. Keyword level
2. Linguistic analysis of phrases level

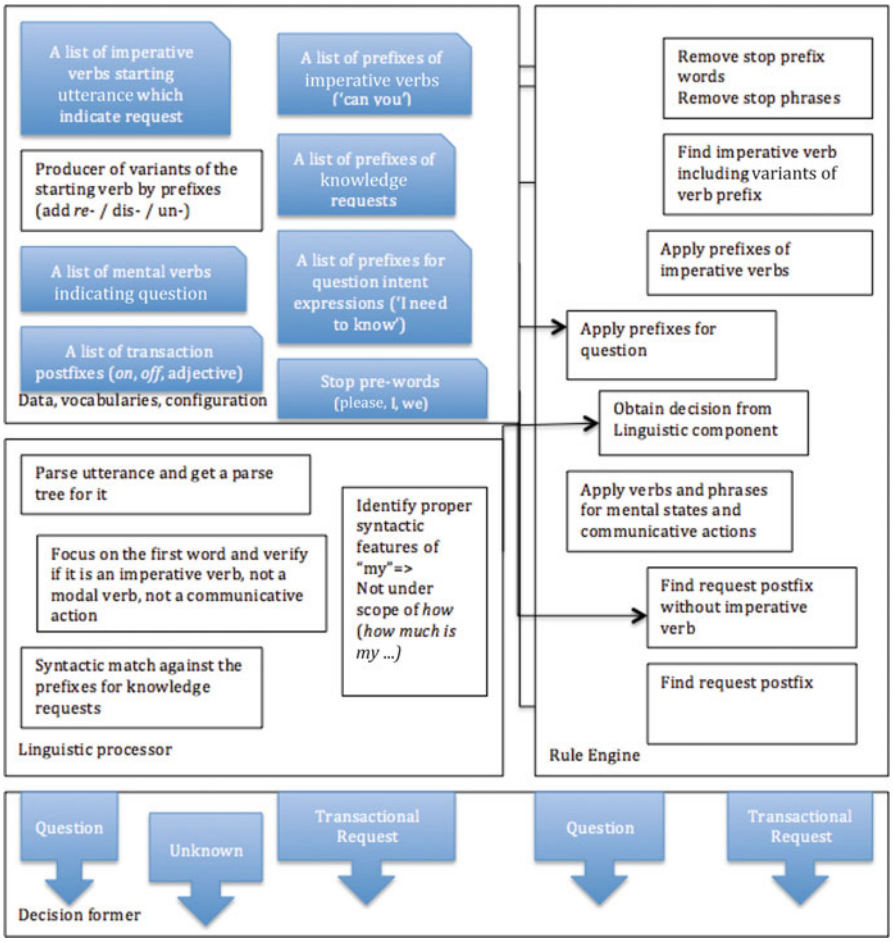

A list of imperative verbs starting utterance which indicate request
A list of prefixes of imperative verbs ('can you')
Producer of variants of the starting verb by prefixes (add re- / dis- / un-)
A list of prefixes of knowledge requests
A list of mental verbs indicating question
A list of prefixes for question intent expressions ('I need to know')
A list of transaction postfixes (on, off, adjective)
Data, vocabularies, configuration
Stop pre-words (please, I, we)
Remove stop prefix words
Remove stop phrases
Find imperative verb including variants of verb prefix
Apply prefixes of imperative verbs
Apply prefixes for question
Obtain decision from Linguistic component
Apply verbs and phrases for mental states and communicative actions
Find request postfix without imperative verb
Find request postfix
Rule Engine
Parse utterance and get a parse tree for it
Focus on the first word and verify if it is an imperative verb, not a modal verb, not a communicative action
Identify proper syntactic features of "my"=> Not under scope of how (how much is my ...)
Syntactic match against the prefixes for knowledge requests
Linguistic processor
Question
Unknown
Transactional Request
Question
Transactional Request
Decision former

Fig. 3.4 The architecture for question vs transaction request recognition. The class labels on the bottom correspond to the decision rules above

The algorithm chart includes four major components (Fig. 3.4):

- Data, vocabularies, configuration
- Rule engine
- Linguistic Processor
- Decision Former

Data, vocabularies, configuration components included leading verbs indicating that an utterance is a request. It also includes expressions used by an utterance author to indicate the he wants something from a peer, such as '*Please do* . . . *for me*'. These expressions also refer to information request such as 'Give me MY . . .' such as account information. For a question, this vocabulary includes the ways people address questions, such as '*please tell me*. . .'.

Rule engine applies a sequence of rules, both keyword-based, vocabulary-based and linguistic. The rules are applied in certain order, oriented to find indication of a transaction. If main cases of *transactions* are not identifies, only then the rule engine applies *question* rules. Finally, if question rules did not fire, we classify the utterance as *unknown,* but nevertheless treat it as default, a *question*. Most rules are specific to the class of requests: if none of them fire then the decision is also a *question*.

Linguistic processor targets two cases: imperative leading verb and a reference to "my" object. Once parsing is done the first word should be a regular verb in present tense, active voice, neither modal, mental (Galitsky 2016) or a form of *be*. These constraints assure this verb is in the imperative form '*Drop the temperature in the room*'. The second case addresses utterance related to an object the author owns or is associated too, such as '*my account balance*' and '*my car*'. These utterances are connected with an intent to perform an action with these objects or request for an information on them (versus a question which expresses a request to share general knowledge, not about this particular, *my* object).

Decision former takes an output of the Rule Engine and outputs one out of three decisions, along with an explanation for each of them. Each fired keyword-based rule provides an explanation, as well as each linguistic rule. So when a resultant decision is produced, there is always a detailed back up of it. If any of the components failed while applying a rule, the resultant decision is *unknown*.

If no decision is made, the chatbot comes back to the user asking for explicit clarification: '*Please be clearer if you are asking a question or requesting a transaction*'.

3.2.1 Examples of Questions and Transactional Requests

We present examples for each class together with the rules which fired and delivered the decision (shown in []).

Questions

If I do not have my Internet Banking User ID and Password, **how can I** login? [*if* and how can I – prefix]
I am **anxious** about spending my money [mental verb]
I am ***worried*** about my spending [mental verb]
I am ***concerned*** about how much I used [mental verb]
I am **intereste***d* how much money I lost on stock [mental verb]
How can **my** saving account be funded [How+my]
Domestic wire transfer [no transactional rule fired therefore question]
order replacement/renewal card not received [no transactional rule fired therefore question]

Transactional Requests

Open iWish – a Flexible Recurring Deposit [leading imperative verb]

Cancel a Fixed Deposit using ICICI Bank VISA Debit Card [leading imperative verb]
Help me to Login and raise a Service Request [leading imperative verb]
Turn the light **on** [postfix]
Put wiper rate on **high** [postfix]
Transfer funds from checking to saving [leading imperative verb]
Move funds from saving to mortgage [leading imperative verb]
Fund my investment account from checking [leading imperative verb + *my* without *How*]
Wire money from **my** checking to investment [leading imperative verb + *my* without *How*]
too loud, quieter please [leading adjective prefix]
set the security system to **off** [postfix]
close the garage door [leading imperative verb]
do western union [leading imperative verb *do*]
give me the check deposited in Bank Account but not credited [leading imperative verb + *me*]

3.2.2 Nearest Neighbor-Based Learning for Questions vs Transactional Requests Recognition

If a chatbot developer/vendor intends to overwrite the default questions vs transactional requests recognition rules, he would need to supply a balanced training set which includes samples for both classes. To implement a nearest-neighbor functionality, we rely on information extraction and search library (Lucene 2018). The training needs to be conducted in advance, but in real time when a new utterance arrives the following happens:

1. An instant index is created from the current utterance;
2. We iterate through all samples from both classes. For each sample, a query is built and search issued against the instant index;
3. We collect the set of queries which delivered non-empty search results with its class and aggregate this set by the classes;
4. We verify that a certain class is highly represented by the aggregated results and the other class has significantly lower presentation. Then we select this highly represented class as a recognition result. Otherwise, the system should refuse to accept a recognition result and issue Unknown.

Lucene default TF*IDF model will assure that the training set elements is the closest in terms of most significant keywords (from the frequency perspective, (Tan 2005; Salton and Yang 1973)). (Trstenjak et al. 2013) present the possibility of using a nearest neighbor (KNN) algorithm with TF*IDF method for text classification.

This method enables classification according to various parameters, measurement and analysis of results. Evaluation of framework was focused on the speed and quality of classification, and testing results showed positive and negative characteristics of TF*IDF-KNN algorithm. Evaluation was performed on several categories of documents in online environment and showed stable and reliable performance. Tests shed the light on the quality of classification and determined which factors have an impact on performance of classification.

3.3 A Decision Support Chatbot

We propose an active learning framework for a decision support where a user expert presents her decision first in an explicit manner. This framework allows the machine learning component leverage the explainable decision presented by the expert first and then produce an explainable decision comparable with the manual one. This type of active learning framework is expected to improve the quality of the resultant, hybrid decision and constantly maintain the decision skill level of the human expert.

In spite of the high success of the existing decision support systems (DSS) (Newman et al. 2000; Hartono et al. 2007; Galitsky et al. 2009), their expert users rely on them more and more, obtain and use a DSS result and become detached from a decision process itself. The issue here is a lack of responsibility of these expert users for the final decision, as well as an accuracy of future decisions (Goldberg et al. 2008). It is well known that a drop of accuracy of DSS system is caused by domain evolution, where the training occurred on the original, old data and the current, new data may significantly deviate. The rate of this domain evolution, concept drift (Krawczyk et al. 2017), can be much higher that the self re-training capabilities of the DSS. Hence it is fairly important that this DSS produces an adequate explanation of its decision so that the human expert can assess this decision with respect to recent data (Shklovskiy-Kordi et al. 2005).

European Union’s new General Data Protection Regulation also control the applicability of machine learning (https://eugdpr.org/). These regulations restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which “significantly affect” users. The law effectively creates a *right to explanation*, whereby a human user can request an explanation of an algorithmic decision that was made about them (Goodman and Flaxman 2017).

In this Section we focus on interactions between a DSS-enabled chatbot and a human expert that goes beyond the active learning framework. For an extended set of features, a decision support session starts with a delivery of a human expert decision to the chatbot, given a set $m << n$ of features this human expert relied his decision upon. Then the chatbot would have to explain disagreement of its decision with the human experts’ prior decision, presenting the set d of features employed in DSS decision.

There are two reasons decision agents including humans and machines make mistakes:

1. recognition model;
2. distortion of parameter values due to measurement errors and cognitive bias.

In most cases, humans are subject to errors associated with their sentiments.

Both machines and humans learn and generalize from data and both make errors (Galitsky and Parnis 2017). (Goldberg et al. 2007) has been investigating this in a medical decision support domain. However, unlike machines, humans possess ontologies and are capable of applying them to assess their decisions. Humans usually learn from a smaller set of examples but the accuracy of their decisions is achieved by applying ontologies on top of generalizations. Even when a machine has an ontology, we assume that it is limited in scope and cannot cover all cases. For humans, all cases are covered by some kind of ontological knowledge.

As an example, we use a case of visual recognition of images of dogs and wolves. Generalizing from available data, machines build a model where if there is a snow in background, it is a wolf. Although this turns out to cover the training set well, human ontology hints that this generalization is prone to errors since dog images can have snow in background as well.

We design dialogue management in our chatbot to facilitate interactions between the DSS and human expert. A sample session is shown in Fig. 3.5.

Meta-agent does not have its own model of a phenomenon; instead, it controls the information exchange between the DSS and human expert in the form of a decision and its explanation as a set of parameters and their values.

3.3.1 Example of a Decision Support Session

We present a classification problem for three animals: a wolf, a dog and a coyote, relying on the following parameters: animal length, skin color, height, speed, tail length and tail direction (Table 3.3).

Human agent and DSS have different models of a phenomenon such as an animal. They cannot exchange model parameters but instead they can encourage each other to pay attention to particular parameters they think are important for recognition.

Step 1. A human expert takes a sample and attempts to solve a problem. Let us imagine the following parameters as identified by her:

Length = 115 sm with the range of possible errors [100–130]
Color = ‘*light grey’* with the range [*white* . . . *grey*]
Height = 70 sm with the range [55–85]
Speed = 40 km/h with the range [35–45]
Tail.length = *long* with the range [*average*]
Tail.direction = *down* with range [*straight*]

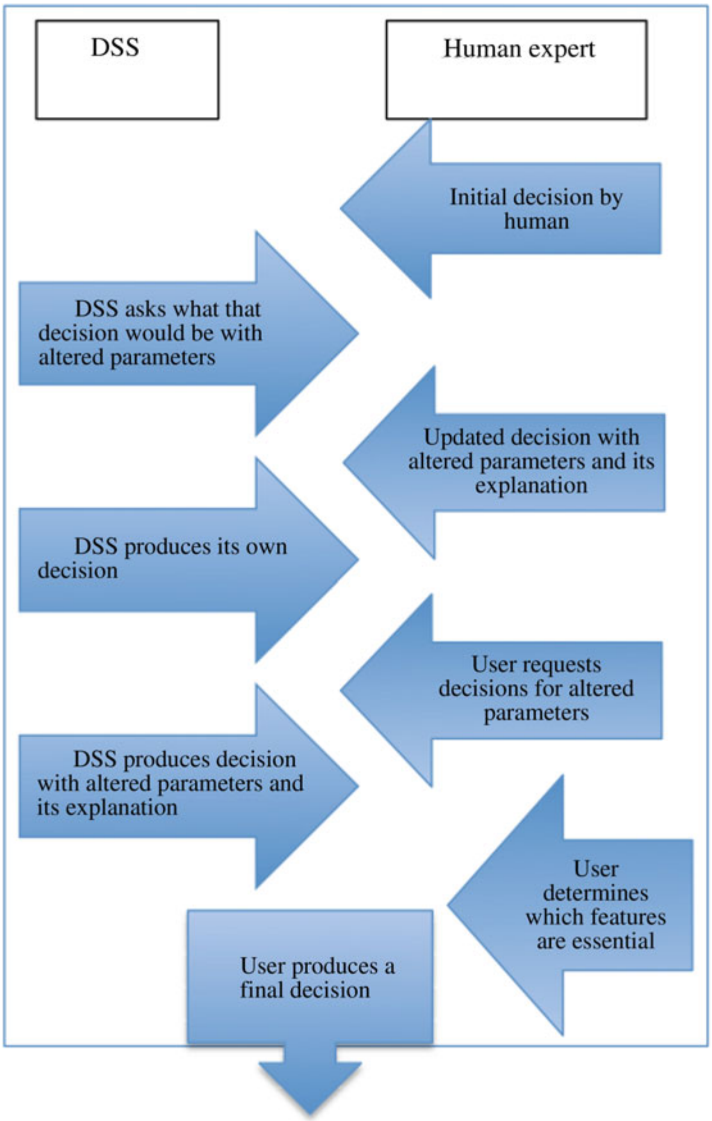

DSS
Human expert
Initial decision by human
DSS asks what that decision would be with altered parameters
Updated decision with altered parameters and its explanation
DSS produces its own decision
User requests decisions for altered parameters
DSS produces decision with altered parameters and its explanation
User determines which features are essential
User produces a final decision

Fig. 3.5 A high-level schema of a dialogue for a DSS chatbot

Step 2. Expert decides that it is a wolf, since (Step 3):

Length = 115
Color = *light grey*
Height = 70 sm
Tail.direction = *down*

Table 3.3 Classification features

	Wolf	Coyote	Greyhound
Length, sm	100–160	75–100	100–120
Height, sm	80–85	45–55	68–76
Color	Gray	Light gray	Any
Speed, km/h	Up to 60	Up to 70	70
Tail length, sm	Long	Average	Long
Tail direction	Down	Down	Not down

Step 4. DSS: If turn length = 115sm into 100sm and height = 70sm into 55 ⇒ *coyote*

If Tail.direction = *straight* ⇒*dog*
If without correction ⇒ *wolf*
Chatbot is asking human about the tail:
Tail.direction = *straight* and Tail.length = *average*, nevertheless ⇒ wolf.
Now the new set of feature values:
Tail.length = *average* with the range [*short*. . .*long*]
Tail.direction = *straight* with range [*down*. . .*up*]

Step 5. DSS ⇒ *dog* since (Step 6)

Tail.direction = *straight*
Speed = 40 km/Ec
(Explanation only for *dog* vs. *wolf*)
Expert: what if Tail.direction = down?
DSS: still *dog* since can only be *wolf*, not *coyote*
Speed = 40 km/h
Tail.length = *average*
Expert: What if both Tail.direction = *down* and speed = 35 km/h?
DSS: then it becomes *wolf*
Expert: What if Tail.direction = *down* and tail.length = *long*?
DSS: *wolf*

Step 7. Now the human expert can do the final judgment.

3.3.2 Computing Decisions with Explanations

Let $\boldsymbol{x} = (x_1, x_2, .., x_n)$ be the n input parameters to the algorithm. x_i can be continuous (numerical) or categorical (Boolean) variables. Let $\boldsymbol{X}$ be a set of $\boldsymbol{x}$. Let $\boldsymbol{v} = (v_1, \ldots, v_n)$ be the particular input values entered by the user.

Let $D = \{\alpha_j\}, j = 1,..., k$ be the set of k possible decisions or output classes.

Let $\alpha_U \in D$ be the initial unassisted decision of the user.

Additionally we allow the user to mark a subset of input parameters $(v_1, \ldots, v_m)$ $m \leq n$ as being particularly important to their decision α_U

We define the decision function f which maps an input vector **v** and a class $\alpha \in D$ to confidence $c \in [0, 1]$:

$$f(\alpha, \boldsymbol{x}) : \alpha, \boldsymbol{x} \to [0, 1].$$

Let α_{ml} be the algorithm decision based on the user-provided input values v.

$$f(\alpha_{ml}, \boldsymbol{v}) = max(f(\alpha, \boldsymbol{x}))\text{for all } \alpha \in \text{D}.$$

For any parameter of $\boldsymbol{x}$, its value x_i may have bias or error so we define $\Omega(x_i)$ such that $\Omega(x_i) > (\Omega(x_i)^{-} \& \ \Omega(x_i) < \Omega\ (x_i)^{+})$ as the set of values which are considered within the error bounds for x_i. The bias includes the uncertainty of an object and uncertainty of the assessor. When there is an uncertainty in assessing a feature, we have the phenomena of "confirmation bias" and "selective perception" (Plous 1993; Lee et al. 2013).

We introduce a feature normalization x_i^{new} for each i-th dimension, set based on the following four thresholds: $a_{0i}, a_{1i}, a_{2i}, a_{3i}, a_{4i}$ (Shklovsky-Kordi et al. 2005):

$$x_i < a_{\mathbf{0i}} : \text{strong\_deviation} : x_i^{new} = 0 + x_i/a_{\mathbf{0i}}$$

$$a_{\mathbf{1i}} < x_i < a_{\mathbf{2i}} : \text{abnormal} : x_i^{new} = 1 + \left(x_i - a_{\mathbf{1j}}\right)/(a_{\mathbf{2i}} - a_{\mathbf{1i}})$$

$$a_{\mathbf{2i}} < x_i < a_{\mathbf{3i}} : \text{normal} : x_i^{new} = 2 + \left(x_i - a_{\mathbf{2j}}\right)/(a_{\mathbf{3i}} - a_{\mathbf{2i}})$$

$$a_{\mathbf{3i}} < x_i < a_{\mathbf{4i}} : \text{abnormal} : x_i^{new} = 3 + \left(x_i - a_{\mathbf{3j}}\right)/(a_{\mathbf{4i}} - a_{\mathbf{3i}})$$

$$\text{a}_{\mathbf{4i}} < \text{x}_i < \text{a}_{\mathbf{4i}} : \text{strong\_deviation} : x_i^{new} = 4 + \text{x}_i/(\text{a}_{\mathbf{4i}})$$

Based on this definition, we compute $\boldsymbol{X} \to \boldsymbol{X}^{new}$ and $\boldsymbol{X}^{new} \to \boldsymbol{X}$. Now we define the similarity between the object $\boldsymbol{x}$ and $\boldsymbol{y}$ as a vector distance $||\boldsymbol{x} - \boldsymbol{y}||$.

Division of the measured value by the accepted average value accomplishes the normalization. The calculation is executed separately for *normal, abnormal* and *strong_deviation* value. To define a range of sub-normal values, a team of experts empirically established the score of acceptable parameters. They are determined for certain combination of features and certain objects. If a parameter stays within the defined abnormal or normal range, no special action is required. The *strong_deviation* range covers all the zone of possible values beyond the abnormal values.

Algorithm for Steps 4: Stability Assessment

Let us consider a n-dimensional space $\Omega(v_1),\ldots, \Omega(v_m), v_{m+1,\ldots,}v_{n.}$ In dimensions 1 to m it is a parallelepiped, and in dimension $m + 1..n$ it is a plane.

Let $\Omega(\boldsymbol{v})$ be set of points where for each *dimension* $\Omega(v_i)^{-} < \Omega(v_i) < \Omega\ (v_i)^{+}$ for $i < m + 1$ and v_i*for* $i > m$.

Let α be the decision of DSS *where* $f(\alpha, \boldsymbol{x}) - f(\alpha_U, \boldsymbol{x}) > 0$ with $\boldsymbol{x} \in \Omega(\boldsymbol{v})$ and $\alpha \in D$. Out of these pairs, let us select the pair $(\alpha_{ml}, \boldsymbol{y})$ which relies on a minimum number of *important* dimensions *1..m*.

Algorithm for Steps 5: Discovering Deviations in Parameters for α_U
The user expert is then suggested to consult parameter i delivering maximum value ($y_i^{new} - v_i^{new}$), $i = 1,..m$, y_i – i-th dimension of vector $\mathbf{y}$, when feature normalization procedure is fixed. If human decision deviates from the DSS decision in initial data, meta-agent needs to focus on a single parameter value from $\{v_{1,\ldots} v_{n,}\}$ that would direct the human expert towards the DSS decision. This is how to find this feature.

What is the *worst* feature dimension for a human decision? To find it we first identify the best feature value (we call it *typical*) for α_U for all i:

$v_i^{typ}(\alpha_U) = max_j\, f(\alpha_U,[v_{1,\ldots,}\, v_{i-1},\, x_j,\, v_{i+1,\ldots,}\, v_n])$ over all values x_j of i-th dimension. For example, $x_1 =$ '*white*', $x_2 =$ '*light grey*',

$$x_3 = \text{'}grey\text{'}, x_4 = \text{'}dark\ grey\text{'}, x_5 = \text{'}black\text{'}, j = 1..5.$$

$$v_i^{typ}(\alpha_U) : color = \text{'}grey\text{'}\ \text{when}\ \alpha_U = \text{'}wolf\text{'}.$$

We do it for all dimensions i.

Now we proceed to the dimension i best for the DSS decision.

$$max_i(f(\alpha_{ml,}[v_{1,\ldots,}v_{i-1,},v_i,v_{i+1,\ldots,}v_n]) - f(\alpha_{ml,}[v_{1,\ldots,}v_{i-1,}v_i^{typ}(\alpha_U),v_{i+1,\ldots,}v_n])$$

Here, the feature could be as follows

$$v_i : color = \text{'}light\ grey\text{'},\ v_i^{typ}(\alpha_U) : color = \text{'}grey\text{'}\ \text{when}\ \alpha_U = \text{'}wolf\text{'}.$$

Algorithm for Step 6: Explainability of DSS
This algorithm attempts to explain the DSS decision for human expert in the same way as has been done by humans. DSS delivers most important features for its decision.

Let us use a random generator with $\mathbf{v}^{new}$ as average value and (1,..,1)-vector as standard deviation to select in X^{new}, where.

$$-\varepsilon < \boldsymbol{f}\ (\alpha_{ml},\boldsymbol{x}) - \boldsymbol{f}\ (\alpha_U,\boldsymbol{x}) < 0$$

Then we take a point **z** delivering the minimum $\| z^{new} - \mathbf{v}^{new} \|$. Then in the cube, we randomly select a point **z'** around **z** in where $-\varepsilon < \boldsymbol{f}(\alpha_{ml}, \boldsymbol{x}) - \boldsymbol{f}(\alpha_U, \boldsymbol{x}) < 0$ such that **z'** gives us a minimum of $\| z'^{new} - \mathbf{v}^{new} \|$. We iteratively set $z = z'$ and do the above iteratively till the distance $\| z'^{new} - \mathbf{v}^{new} \|$ stops decreasing.

Feature i which do not belong to $\Omega(z'_i)$ is important for decision making of DSS to obtain the decision α_{ml} that different from α_U. Most important features i are those where $z_i^{new} - v_i^{new}) >= 1$.

Here is the user interaction flow (Fig. 3.6):

1st Step:
User input: $\mathbf{v} = [v_1, \ldots, v_n] \in \boldsymbol{X}$.

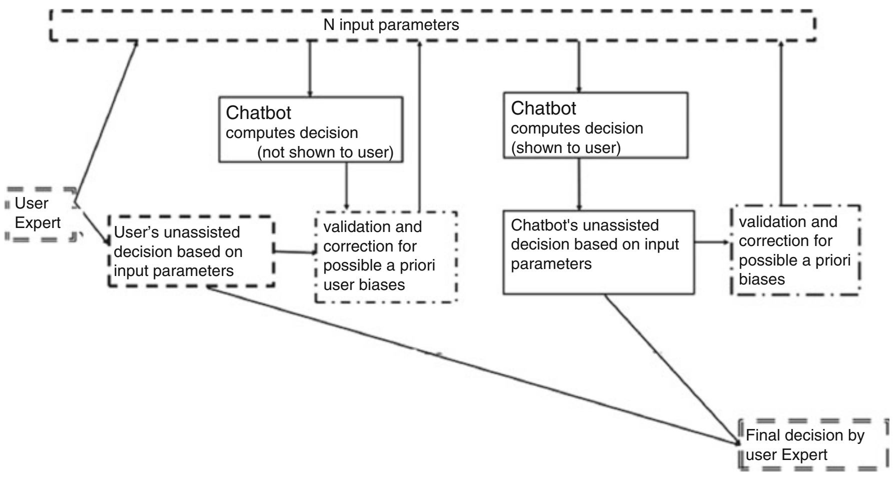

N input parameters
Chatbot computes decision (not shown to user)
Chatbot computes decision (shown to user)
User Expert
User's unassisted decision based on input parameters
validation and correction for possible a priori user biases
Chatbot's unassisted decision based on input parameters
validation and correction for possible a priori biases
Final decision by user Expert

Fig. 3.6 Architecture of a decision support chatbot

2nd Step:
Initial unassisted decision α_U of the user

3nd Step:
User indicates m out of n input values as being particularly important to his decision $\alpha_U\ [v_{1,\ldots}\ v_m]\ m << n$

4th Step:
Now DSS verified the decisions of user α_U without sharing $\alpha_{ml.}$
In order to determine how stable α_U is relatively to perturbations of $\boldsymbol{v}$ within error bounds Ω, we compute α_{ml} by means of Stability Assessment Algorithm.
If α_{ml} does not match α_U, go to Step 5.
If α_{ml} matches α_U, then α_U is selected as a preliminary solution, and we proceed to Step 6.
5th Step:
Since $\alpha_U \neq \alpha_{ml}$ we iteratively work with the user to see if we can converge on a stable decision. We apply Discovering abnormal parameters Algorithm.

We could, at this point, just show α_{ml} to the user, but we specifically avoid doing this in order to prevent the user from unthinkingly changing their decision to α_{ml}. Instead we use a more nuanced, indirect approach where we try to find the parameter whose value $v_{i,}$ from the ones indicated by the user to be in the set proving $\alpha_{U,}v_{i,}$ is such that its possible deviation affects α_U in the highest degree.

After finding this parameter, we report to the user that the value they provided for this parameter is to some degree inconsistent with α_U. We then give the user the option to change their initial $\alpha_{U.}$

If the user maintains the same decision α_U, α_U is set as a preliminary decision and we proceed go to Step 6.

If user changes their decision, go to Step 2 (unless this point is reached a third time, in which case go to Step 6 to avoid an overly long interaction loop).

6th Step:

Compute decision α_{ml} based on unchanged input values $f(\alpha_{ml}, v)$. α_{ml} is set as a decision of DSS and is shown to the human expert along with the set of key features which has yielded α_{ml} instead of α_U. Explainability of DSS algorithm is in use here.

7th Step:

The human expert can modify v and observe respective decisions of DSS. DSS can in turn change its decision, and provide an updated explanation. Once the human expert obtained DSS decision for all cases of interest, she obtains the final decision.

Hence in the third step the human explains its decision, and in the sixth step the DSS explains its decision. In the fifth step DSS assesses the stability of human experts' decision with respect to selected features. In the seventh step the human expert does the same with DSS decisions. So the sixth step is inverse to the third and the seventh is inverse to the fifth.

Hence we constructed a goal –oriented chatbot that instead of answering factual questions or performing transactions conducts a decision support.

For a chatbot to handle explainable decision support, explanation format should be simple and have a natural representation, as well as match the intuition of a human expert. Also, it should be easy to assess DSS explanation stability with respect to deviation of decision features. Available methods such as (Baehrens et al. 2010) where DSS is a black box, similar to the current setting, do not obey all of these requirements.

3.4 Explanation-Based Learning System *Jasmine*

We describe a deterministic inductive learning system *Jasmine* configured to predict an utterance in a dialogue. This system belongs to the class of Inductive Logic Programming and Explanation Based learning systems operating with data which is fully formalized and made sense of. Originally, *Jasmine* was designed to support learning in a number of domains which require learning *explainability*: not just a correct classification or prediction needs to be made, but also it needs to be explained in terms of which samples and which rules were employed in the decision (Galitsky et al. 2007).

The foundation of learning and cognition is an inductive reasoning pattern. If we want to recognize a specific kind of dialogs and distinguish it from other kinds, we want to find a common feature (phrase, entity) shared by at least two dialog instances. This feature should not be present in other kinds of dialogs. The principle of induction states that a commonality of features between the patterns (such as ‘deny responsibility’ response) causes the target feature (the utterance that includes a ‘*threat of proceeding to a Better Business Bureau claim*’). This principle is referred to as the *direct method of agreement*.

If two or more instances of the phenomenon under investigation have only one circumstance in common, the circumstance in which alone all the instances agree, is the cause (or effect) of the given phenomenon (Mills 1843).

3.4.1 A Reasoning Schema

Jasmine is based on a learning model called JSM-method (Anshakov et al. 1989, in honor of John Stuart Mill, the English philosopher who proposed schemes of inductive reasoning in the nineteenth century). JSM-method to be presented in this section implements Mill’s direct method of agreement stating that similar effects (associated features, target features) are likely to follow common causes (attributes), as well as abduction in the form of explainability. JSM attempts to solve the problem of *inductive bias*, a means to select one generalization over another. It is hard for an automated learning system to find a proper generalization level, making decisions in the real world.

The task of Jasmine is to predict or recognize a *target feature* (phrases in a dialog utterance to follow) given the observable features (phrases and entities in the previous utterances). These features are observed in the *objects* of a training set so that a target feature of new, unknown object can be recognized or predicted (Galitsky et al. 2007).

Given the *features* of *objects of a training set*, we intend to obtain *a logical expression* for the target feature that includes all positive examples and excludes all negative examples, given some initial formalized background knowledge. In the Explanation-based Learning setting such expression for the target feature is a logical consequence of background knowledge and training dataset; however, this condition is not always viable in a domain of human learning from with experimental observations. Explanation-based Learning is designed to generalize form a single example; however, in human learning domains one would prefer more reliable conclusions from multiple observations. These multiple observations (examples) may introduce inconsistencies; and the desired machine learning technique should be capable of finding consistent explanations linking possibly mutually inconsistent observations with the target feature.

Within *Jasmine* first-order language, objects are atoms, and known features and the target feature are the terms which include these atoms. For a given target feature, a term for a feature of an object can be as follows:

- *Positive*
- *Negative*
- *Inconsistent*
- *Unknown*

In our case, building (predicting) a dialog step, objects are utterances and object features are phrases of these utterances.

An inference to obtain this target feature (satisfied or not) can be represented as one in a respective four-valued logic (Anshakov et al. 1989). The predictive machinery is based on building hypotheses in the form of clauses

$target\_feature(O)\text{:-}\ feature_1(O, \ldots), \ldots, feature_n(O, \ldots),$ that separate examples,

where *target_feature(O)* is to be predicted, and $features_1, \ldots, feature_n \in features$ are the features the target feature is associated with; *O* ranges over objects.

Desired separation is based on the *similarity* of objects in terms of features they satisfy (according to the direct method of agreement above). Usually, such similarity is domain-dependent. However, building the general framework of inductive-based prediction, we use the anti-unification of formulas that express the totality of features of the given and other objects (our features (causes) do not have to be unary predicates; they are expressed by arbitrary first-order terms). We assume the human learning to be as general and flexible as this operation of anti-unification, to be introduced.

Figure 3.7 is an example of a learning setting, where features, objects, the target feature and the knowledge base are given. We keep using the conventional PROLOG notations for variables and constants.

In a numerical, statistical learning similarity between objects is expressed by a number. In deterministic, structured learning with explainability of results similarity is a *structure.* Similarity between a pair of objects is a hypothetical object which obeys the common features of this pair of objects. In handling similarity *Jasmine* is close to Formal Concept Analysis (Ganter and Wille 1999; Ganter and Kuznetsov

```
features([e1, e2, e3, e4, e5, e6, oa1, oa2, ap1, ap2, ap3, ap4, f1, f2,
  cc4, cc5, cc6, cc7, cb5, cb7]). % dialog phrases
objects([o1, o2, o3, o4, o5, o6, o7, o8, o9, o10, o11, o12, o13, o14,
  o15, o16, o17, o18]).  % utterances
target_feature [cb5]).  % a phrase to be included in an utterance
   % Beginning of knowledge base %
e1(o1). oa1(o1). ap1(o1). ap3(o1). f1(o1).  cc5(o1). cb5(o1).
e1(o2). oa1(o2). ap1(o2). ap3(o2). f1(o2).  cc5(o2). cb5(o2).
e2(o8). oa2(o8). ap2(o8). ap1(o8). f1(o8).  cc5(o8). cb5(o8).
e3(o10). oa1(o10). a3(o10). ap2(o10). f1(o10). cc4(o10).
e3(o11). oa1(o11). a3(o11). ap2(o11). f1(o11). cc4(o11). cb5(o11). cb7(o11).
e4(o16). oa1(o16). a1(o16). ap1(o16). f1(o16). cc5(o16). cb5(o16).
e5(o17). oa1(o17). a4(o17). ap2(o17). f1(o17). cc6(o17). cb7(o17).
e6(o18). oa1(o18). a1(o18). ap2(o18). f1(o18). cc4(o18). cb7(o18).
   %% End of knowledge base
unknown(cb5(o10)). % Should the current utterance, o10, include the phrase
'cb5'?
```

Fig. 3.7 A sample knowledge base for high-level mining of protein sequence data

2001), where similarity is the *meet* operation of a *lattice* (called concept lattice) where features are represented by unary predicates only. For the arbitrary first-order formulas for objects in *Jasmine* we choose the anti-unification of formulas which expresses features of the pair of objects to derive a formula for similarity sub-object (Chap. 5, Galitsky 2017). Below we will be using the predicate

similar(Object1, Object2, CommonSubObject) which yields the third argument given the first and the second arguments.

The reasoning procedure of *Jasmine* is shown in Fig. 3.8. Note that the prediction schema is oriented to discover which features cause the target feature and how (the causal link) rather than just searching for common features for the target feature (which would be much simpler, 6 units on the top). The respective clauses (1–4) and sample results for each numbered unit (1–4) are presented in Fig. 3.9.

Let us build a framework for predicting the target feature *V* of objects set by the formulas *X* expressing their features: *unknown(X, V)*. We are going to predict whether $V(x_1, \ldots, x_n)$ holds or not, where $x_1, \ldots, x_n$ are variables of the formula set *X* (in our example, $X = \mathrm{cb5(o10)}$, $x_1 = \mathrm{o10}$).

We start with the raw data, positive and negative examples, raw*Pos(X, V)* and *rawNeg(X, V)*, for the target feature *V*, where *X* range over formulas expressing features of objects. We form the totality of intersections for these examples (positive ones, *U*, that satisfy *iPos(U,V)*, and negative ones, *W*, that satisfy *iNeg (W,V),* not shown):

$$iPos(U, V) : -rawPos(X1, V), rawPos(X2, V), X1\backslash = X2, similar(X1, X2, U), U\backslash = [\,]. \tag{3.1}$$

$iPos(U, V) : - iPos(U1, V), rawPos(X1, V), similar(X1, U1, U), U\backslash = [\,]$. Above are the recursive definitions of the intersections. As the logic program clauses which actually construct the lattice for the totality of intersections for positive and negative examples, we introduce the third argument to accumulate the currently obtained intersections (the negative case is analogous):

$$iPos(U, V) : -iPos(U, V, \_).$$

$$\begin{aligned} iPos(U, V, Accums) : -\ & rawPos(X1, V),\ rawPos(X2, V),\ X1\backslash = X2,\ similar(X1, \\ & X2, U),\ Accums = [X1, X2],\ U\backslash = [\,]. \end{aligned}$$

$$\begin{aligned} iPos(U, V, AccumsX1) : -\ & iPos(U1, V, Accums),\ !\ ,\ rawPos(X1, V),\ not\ member \\ & (X1, Accums),\ similar(X1,\ U1,\ U),\ U\backslash = [],\ append \\ & (Accums, [X1], AccumsX1). \end{aligned}$$

As one can see, there is a “symmetric” treatment of positive and negative examples and hypotheses: *Jasmine* uses negative examples to falsify hypotheses that have counter-examples. On the contrary, a simplified Explanation-based Learning uses only positive examples and can be viewed as just the left side of Figure 3.8.

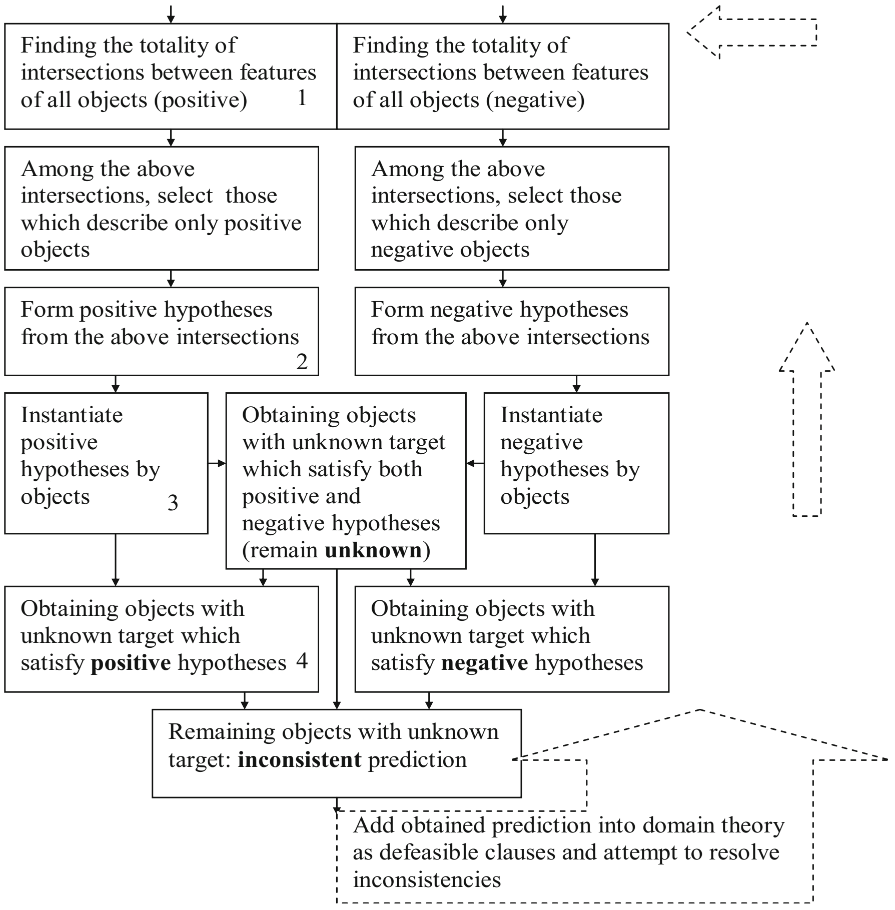

Finding the totality of intersections between features of all objects (positive) 1
Finding the totality of intersections between features of all objects (negative)
Among the above intersections, select those which describe only positive objects
Among the above intersections, select those which describe only negative objects
Form positive hypotheses from the above intersections 2
Form negative hypotheses from the above intersections
Instantiate positive hypotheses by objects 3
Obtaining objects with unknown target which satisfy both positive and negative hypotheses (remain unknown)
Instantiate negative hypotheses by objects
Obtaining objects with unknown target which satisfy positive hypotheses 4
Obtaining objects with unknown target which satisfy negative hypotheses
Remaining objects with unknown target: inconsistent prediction
Add obtained prediction into domain theory as defeasible clauses and attempt to resolve inconsistencies

Fig. 3.8 The chart for reasoning procedure of *Jasmine*

To obtain the actual positive *posHyp* and negative *negHyp* hypotheses from the intersections derived above, we filter out the inconsistent hypotheses which belong to both positive and negative intersections *inconsHyp(U, V)*:

$$inconsHyp(U, V) : -iPos(U, V), iNeg(U, V). \tag{3.2}$$

$$posHyp(U, V) : -iPos(U, V), not\ inconsHyp(U, V).$$

$$negHyp(U, V) : -\ iNeg(U, V),\ not\ inconsHyp(U, V).$$

1. Intersections
Positive: [[e1(_),oa1(_),ap1(_),ap3(_),f1(_),cc5(_)],
[ap1(_),f1(_),cc5(_)],[ap1(_),f1(_)],[oa1(_),f1(_)], [oa1(_),ap1(_),f1(_),cc5(_)],
[e2(_),e3(_),oa2(_),ap1(_),ap2(_),f1(_)],[e3(_),ap2(_),f1(_)],[e4(_),oa1(_),ap1(
_),
f1(_),cc5(_)]]
Negative: [[oa1(_),ap2(_),f1(_),cb7(_)]]
Unassigned examples:

2. Hypotheses
Positive:[e1(_),oa1(_),ap1(_),ap3(_),f1(_),cc5(_)],[ap1(_),f1(_),cc5(_)],
[ap1(_),f1(_)],[oa1(_),f1(_)],[oa1(_),ap1(_),f1(_),cc5(_3B60)],
[e2(_),e3(_),oa2(_),ap1(_),ap2(_),f1(_)], [e3(_),ap2(_),f1(_)],
[e4(_),oa1(_),ap1(_),f1(_),cc5(_)]]
Negative: [[oa1(_),ap2(_),f1(_),cb7(_)]]
Contradicting hypotheses: []
The clauses for hypotheses here are:
cb5(X)>-
e1(X),oa1(X),ap1(X),ap3(X),f1(X),cc5(X);ap1(X),f1(X),cc5(X);ap1(X),f1(X).
cb5(X)>- not (oa1(X),ap2(X),f1(X),cb7(X)). Note that all intersections are turned into hypotheses because there is no overlap between positive and negative ones

3. Background (positive and negative objects with respect to the target feature cb5)
Positive:
[[e1(o1),oa1(o1),ap1(o1),ap3(o1),f1(o1),cc5(o1)],[e1(o2),oa1(o2),ap1(o2),ap3(
o2),f1(o2),cc5(o2)],
[e2(o7),e3(o7),oa2(o7), ap1(o7),ap2(o7),f1(o7),cc5(o7)],
[e2(o8),e3(o8),oa2(o8),ap1(o8),ap2(o8),f1(o8)],
[e3(o11),oa1(o11),ap2(o11),f1(o11),cc4(o11),cb7(o11)],[e4(o15),
oa1(o15),ap1(o15),f1(o15),cc5(o15)],
[e4(o16),oa1(o16),ap1(o16),f1(o16),cc5(o16)]]
Negative:
[[e5(o17),oa1(o17),ap2(o17),f1(o17),cc6(o17),cb7(o17)],[e6(o18),oa1(o18),ap
2(o18),f1(o18),cc4(o18),cb7(o18)]]
Inconsistent: []

4. Prediction for cb5 (objects o10)
Positive: [[e3(o10),oa1(o10),ap2(o10),f1(o10),cc4(o10)]]
Negative:[]
Inconsistent: []
Uninstantiated derived rules (confirmed hypotheses)
cb5(O):- e3(O), oa1(O),ap2(O), f1(O), cc4(O).

Fig. 3.9 The *Jasmine* prediction protocol. Steps are numbered in accordance to the units in Fig. 3.8

Here *U* is the formula expressing the features of objects. It serves as a body of clauses for hypotheses *V*: – *U*.

The following clauses deliver the totality of objects so that the features expressed by the hypotheses are *included* in the features of these objects. We derive positive and negative hypotheses *reprObjectsPos(X, V)* and *reprObjectsNeg(X, V)* where *X* is instantiated with objects (*V* is positive and negative respectively). The last clause (with the head *reprObjectsIncons(X, V)*) implements the search for the objects to be predicted so that the features expressed by both the positive and negative hypotheses are included in the features of these objects.

$$reprObjectsPos(X, V) : -rawPos(X, V), posHyp(U, V), similar(X, U, U). \quad (3.3)$$

$$\begin{aligned} reprObjectsNeg\ (X, V) :\ &-\ rawNeg(X, V),\ negHyp(U, V),\ similar(X, U, U).\\ reprObjectsIncons(X, V) :\ &-\ unknown(X, V),\ posHyp(U1, V),\ negHyp(U2, V),\\ &similar(X, U1, U1),\ similar(X, U2, U2). \end{aligned}$$

Two clauses above (top and middle) do not participate in prediction directly; their role is to indicate which objects deliver what kind of prediction.

Finally, we approach the clauses for prediction. For the objects with unknown target features, the system predicts that they either satisfy these target features, do not satisfy these target features, or that the fact of satisfaction is inconsistent with the raw facts. To deliver *V*, a positive hypothesis has to be found so that the set of features *X* of an object has to include the features expressed by this hypothesis, and *X* should not be from *reprObjectsIncons(X, V).* To deliver $\neg V$, a negative hypothesis has to be found so that a set of features *X* of an object has to include the features expressed by this hypothesis and *X* is not from *reprObjectsIncons(X, V).* No prediction can be made for the objects with features expressed by *X* from the third clause,
predictIncons(X,V).

$$\begin{aligned} \boldsymbol{predictPos(X, V){:}-unknown(X, V), posHyp(U, V), similar(X, U, U),}\\ \boldsymbol{not\ reprObjectsIncons(X, V).} \end{aligned} \quad (3.4)$$

$$\begin{aligned} predictNeg(X, V) : -unknown(X, V), negHyp(U, V), similar(X, U, U),\\ not\ reprObjectsIncons(X, V). \end{aligned}$$

$$\begin{aligned} predictIncons(X, V) : -unknown(X, V), not\ predictPos(X, V),\\ not\ predictNeg(X, V),\ not\ reprObjectsIncons(X, V). \end{aligned}$$

The first clause above (shown in bold) will serve as an entry point to predict (choose) a given target feature among a generated list of possible target features that can be obtained for the current state. The clause below is an entry point to *Jasmine* if it is integrated with other applications and/or reasoning components.

predict_target_feature_by_learning(GoalConceptToBePredicted,S):-.
findAllPossibleGoalConcepts (S, As), *loadRequiredSamples*(As),
member$(EffectToBePredicted, As)$,
predictPos$(X, GoalConceptToBePredicted), X\backslash = [\,]$.

Predicate *loadRequiredSamples(As)* above forms the training dataset. If for a given dataset a prediction is inconsistent, it is worth eliminating the objects from the dataset which deliver this inconsistency. Conversely, if there are an insufficient number of positive or negative objects, additional ones are included in the dataset. A number of iterations may be required to obtain a prediction, however the iteration procedure is monotonic and deterministic: the source of inconsistency/insufficient data cases are explicitly indicated at the step where predicates *reprObjectsPos* and *reprObjectsNeg* introduced above are satisfied. This is the solution to the so called *blame assignment* problem, where by starting at the erroneous or inconsistent conclusion and tracking backward through the explanation structure, it is possible to identify pieces of domain knowledge that might have caused an error or inconsistency (Galitsky et al. 2007).

When the set of obtained rules *posHyp* and *negHyp* for positive and negative examples (together with the original domain theory) is applied to a more extensive (evaluation or exploration) dataset, some of these rules may not always hold. If at the first run 1)-4) *Jasmine* refuses to make predictions for some objects with unknown target features, then a repetitive iteration may be required, attempting to use newly generated predictions to obtain objects' target features which are currently unavailable. The arrows on the right of Fig. 3.8 illustrate this kind of iterative process.

For example, for the knowledge base Fig. 3.7 above, we have the following protocol and results (Fig. 3.9):

Hence *cb5(o10)* holds, which means that the sequence o10 has the length of loop of 5 amino acids.

3.4.2 *Computing Similarity Between Objects*

The quality of *Jasmine*-based prediction is dramatically dependent on how the similarity of objects is defined. Usually, high prediction accuracy can be achieved if the measure of similarity is sensitive to object features which determine the target feature (explicitly or implicitly, Galitsky and Shpitsberg 2016). Since most of times it is unclear in advance which features affect the target feature, the similarity measure should take into account all available features. If the totality of selected features describing each object is expressed by formulas, a reasonable expression of similarity between a pair of objects is the following. It is a formula that is the least common generalization of the formulas for both objects, which is anti-unification, mentioned in the previous section. Anti-unification is the inverse operation to the unification of formulas in logic programming. Unification is the basic operation which finds the least general (instantiated) formula (if it exists), given a pair of formulas. Anti-unification was used in as a method of generalization; later this work was extended to form a theory of inductive generalization and hypothesis formation. Anti-unification, in the finite term case, was studied as the least upper bound operation in a lattice of terms.

For example, for two formulas *p(a, X, f(X))* and *p(Y, f(b), f(f(b)))* their anti-unification (least general generalization) is *p(Z1, Z2, f(Z2))*. Conversely, unification of this formulas, *p(a, X, f(X))* = *p(Y, f(b), f(f(b)))* will be *p(a, f(b), f(f(b)))*. Our logic programming implementation of anti-unification for a pair of conjunctions, which can be customized to a particular knowledge domain, is presented in Fig. 3.10.

Although the issue of implementation of the anti-unification has been addressed in the literature, we present the full code to have this book self-contained (Galitsky 2014). In a given domain, additional constraints on terms can be enforced to express a domain-specific similarity. Particularly, certain arguments can be treated differently (should not be allowed to change if they are very important, or should form a special kind of constant). A domain – specific code should occur in the line shown in bold.

There are other *Jasmine*-compatible approaches to computing similarities except the anti-unification. In particular, it is worth mentioning the graph-based approach of finding similarities between scenarios. The operation of finding the maximum common subgraphs serves the purpose of anti-unification in such the domain (Chap. 5, Galitsky 2015). This operation was subject to further refinement expressing similarities between scenarios of multiagent interaction, where it is quite important to take into account different roles of edges of distinct sorts.

Novice users of *Jasmine* are advised to start building the similarity operation as an intersection between objects' features (unordered set of features) and obtain an initial prediction. Then, when the explanations for predictions are observed, the users may feel that less important features occur in these explanations too frequently, and anti-unification expression should be introduced so that less important features are nested deeper into the expressions for objects' features. Another option is to build a domain-specific Prolog predicate that computes unification, introducing explicit conditions for selected variables (bold line in the Fig. 3.10).

3.5 Conclusions

The ML community expects to see more deep learning models whose internal memory (bottom-up knowledge learned from the data) is enriched with an external memory (top-down knowledge inherited from a thesaurus). Integrating symbolic (explainable) and sub-symbolic (partially-explainable) AI will be a key towards natural language understanding. Relying on ML is fruitful to make a good guess employing the past experience, because sub-symbolic methods encode correlation and their decision-making process is probabilistic (Young et al. 2018). Natural language understanding, however, requires much more than that. According to Noam Chomsky, "you do not get discoveries in the sciences by taking huge amounts of data, throwing them into a computer and doing statistical analysis of them: that's not the way you understand things, you have to have theoretical insights".

```
similar(F1, F2, F):- antiUnifyFormulas(F1, F2, F).
   antiUnifyFormulas(F1, F2, F):- clause_list(F1, F1s), clause_list(F2, F2s),
     findall( Fm, (member(T1, F1s), member(T2, F2s),
         antiUnifyTerms(T1, T2, Fm)), Fms), %finding pairs
        %Now it is necessary to sort out formulas which are not
        % most general within the list
     findall( Fmost, (member(Fmost, Fms),
        not ( member(Fcover, Fms), Fcover \= Fmost,
            antiUnifyTerms(Fmost, Fcover, Fcover)) ), Fss),
       clause_list(F, Fss). % converting back to clause

antiUnifyTerms(Term1, Term2,Term):-
     Term1=..[Pred0|Args1],len(Args1, LA),% make sure predicates
    Term2=..[Pred0|Args2],len(Args2, LA),% have the same arity
    findall( Var, (   member(N,   [0,1,2,3,4,5,6,7,8,9,10 ]), % not more than 10
arguments
     [! sublist(N, 1, Args1, [VarN1]), %loop through arguments
          sublist(N, 1, Args2, [VarN2]),
      string_term(Nstr,N), VarN1=..[Name|_],         string_term(Tstr,Name),
      concat(['z',Nstr,Tstr],ZNstr),    atom_string(ZN, ZNstr) !],
          % building a canonical argument to create a variable
          % as a result of anti-unification
    ifthenelse( not (VarN1=VarN2),
    ifthenelse(( VarN1=..[Pred,_|_],VarN2=..[Pred,_|_]),
          ifthenelse( antiUnifyConst(VarN1, VarN2, VarN12),
           %going deeper into a subterm when an argument is a term
              (Var=VarN12),     Var=ZNstr) ),
 %OR domain-specific code here for special treatment of certain arguments
 % various cases: variable vs variable, or vs constant, or constant vs constant
          Var=ZNstr),Var=VarN1)             ), Args),
          Term=..[Pred0|Args].
```

Fig. 3.10 The clauses for logic program for anti-unification (least general generalization) of two formulas (conjunctions of terms). Predicate *antiUnify(T1, T2, Tv)* inputs two formulas (scenarios in our case) and outputs a resultant anti-unification

In this chapter we focused on explainable machine learning, which has become less popular in comparison with statistical and deep learning approaches frequently thought of as central in modern AI. We demonstrated that unlike the academic community of machine learners, end users strongly prefer explainable AI. We focused on a chatbot focused on explaining its decisions, presented a logic programming based ML framework and conclude that it is beneficial for a chatbot to perform its dialogue management relying on an explainable ML. We will draw a further comparison on statistical and rule-based methods in Chap. 4 in relation to a NL access to a database, one of the essential chatbot skills.

References

Anshakov OM, Finn VK, Skvortsov DP (1989) On axiomatization of many-valued logics associated with formalization of plausible reasoning. Stud Logica 42(4):423–447

Arras L, Horn F, Montavon G, Müller K-R, Samek W (2017) What is relevant in a text document?: an interpretable machine learning approach. PLoS One. https://doi.org/10.1371/journal.pone.0181142

Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, Müller K-R (2010) How to explain individual classification decisions. J Mach Learn Res 11(June):1803–1831

DARPA (2016) Explainable artificial intelligence (XAI). http://www.darpa.mil/program/explainable-artificial-intelligence. Last downloaded November 2018

Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. Eng Appl Artif Intell 32:160–184

Galitsky B (2015) Finding a lattice of needles in a haystack: forming a query from a set of items of interest. FCA4AI@ IJCAI, pp 99–106

Galitsky B (2016) Theory of mind engine. In: Computational autism. Springer, Cham

Galitsky B (2017) Matching parse thickets for open domain question answering. Data Knowl Eng 107:24–50

Galitsky B, de la Rosa JL (2011) Concept-based learning of human behavior for customer relationship management. Inf Sci 181(10):2016–2035

Galitsky BA, Ilvovsky D (2017) Chatbot with a discourse structure-driven dialogue management. EACL Demo E17–3022. Valencia

Galitsky B, Parnis A (2017) How children with autism and machines learn to interact. In: Autonomy and artificial intelligence: a threat or savior. Springer, Cham

Galitsky B, Shpitsberg I (2016) Autistic learning and cognition. In: Computational autism. Springer, Cham, pp 245–293

Galitsky B, Kuznetsov SO, Vinogradov DV (2007) Applying hybrid reasoning to mine for associative features in biological data. J Biomed Inform 40(3):203–220

Galitsky B, González MP, Chesñevar CI (2009) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. Decis Support Syst 46 (3):717–729

Ganter B, Kuznetsov S (2001) Pattern structures and their projections. In: Stumme G, Delugach H (eds) Proceedings of the 9th international conference on conceptual structures, ICCS’01. Lecture Notes in Artificial Intelligence, 2120, pp 129–142

Ganter B, Wille R (1999) Formal concept analysis: mathematical foundations. Springer, Berlin

Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L (2018) Explaining explanations: an approach to evaluating interpretability of machine learning. https://arxiv.org/pdf/1806.00069.pdf

Goldberg S, Shklovskiy-Kordi N, Zingerman B (2007) Time-oriented multi-image case history – way to the “disease image” analysis. VISAPP (Special Sessions):200–203

Goldberg S, Niemierko A, Turchin A (2008) Analysis of data errors in clinical research databases. AMIA Annu Symp Proc 6:242–246

Goodman B, Flaxman S (2017) European Union regulations on algorithmic decision-making and a “right to explanation”. AI Mag 38(3):50–57

Hartono E, Santhanam R, Holsapple CW (2007) Factors that contribute to management support system success: an analysis of field studies. Decis Support Syst 43(1):256–268

Krakovna V, Doshi-Velez F (2016) Increasing the interpretability of recurrent neural networks using hidden markov models. CoRR. abs/1606.05320

Krawczyk B, Minku LL, Gama J, Stefanowski J, Wozniak M (2017) Ensemble learning for data stream analysis: a survey. Inf Fusion 37:132–156

Lake BM, Salakhutdinov R, Tenenbaum JB (2015) Human-level concept learning through probabilistic program induction. Science 350(6266):1332–1338

Lee CJ, Sugimoto CR, Zhang G, Cronin B (2013) Bias in peer review. J Am Soc Inf Sci Tec 64:2–17

Liu M, Shi J, Li Z, Li C, Zhu J, Liu S (2017) Towards better analysis of deep convolutional neural networks. IEEE Trans Vis Comput Graph 23(1):91–100

Mann W, Thompson S (1988) Rhetorical structure theory: towards a functional theory of text organization. Text-Interdiscip J Stud Discourse 8(3):243–281

Mill JS (1843) A System of Logic 1843. Also available from University Press of the Pacific, Honolulu, 2002

Newman S, Lynch T, A Plummer A (2000) Success and failure of decision support systems: learning as we go. J Anim Sci 77:1–12

Plous S (1993) The psychology of judgment and decision making, p 233

Salton G, Yang CS (1973) On the specification of term values in automatic indexing. J Doc 29:351–372

Shklovskiy-Kordi N, Shakin VV, Ptashko GO, Surin M, Zingerman B, Goldberg S, Krol M (2005) Decision support system using multimedia case history quantitative comparison and multivariate statistical analysis. CBMS:128–133

Shklovsky-Kordi N, Zingerman B, Rivkind N, Goldberg S, Davis S, Varticovski L, Krol M, Kremenetzkaia AM, Vorobiev A, Serebriyskiy I (2005) Computerized case history – an effective tool for Management of Patients and Clinical Trials. In: Engelbrecht R et al (eds) Connecting medical informatics and bio-informatics, vol 2005. ENMI, pp 53–57

Tan S (2005) Neighbor-weighted K-nearest neighbor for unbalanced text corpus. Expert Syst Appl 28:667–671

Trstenjak B, Sasa M, Donko D (2013) KNN with TF-IDF based framework for text categorization. Procedia Eng 69:1356–1364

Young T, Devamanyu Hazarika, Soujanya Poria, Erik Cambria (2018) Recent trends in deep learning based natural language processing. https://arxiv.org/pdf/1708.02709.pdf