The Thirty-First AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-19)

Probabilistic-Logic Bots for Efficient Evaluation of Business Rules Using Conversational Interfaces

Joseph Bockhorst,[1] **Devin Conathan,**[1] **Glenn M Fung**[1]
[1]American Family Insurance
jbockhor@amfam.com, dconathan@amfam.com, gfung@amfam.com

Abstract

We present an approach for designing conversational interfaces (chatbots) that users interact with to determine whether or not a business rule applies in a context possessing uncertainty (from the point of view of the chatbot) as to the value of input facts. Our approach relies on Bayesian network models that bring together a business rule's logical, deterministic aspects with its probabilistic components in a common framework. Our probabilistic-logic bots (PL-bots) evaluate business rules by iteratively prompting users to provide the values of unknown facts. The order facts are solicited is dynamic, depends on known facts, and is chosen using mutual information as a heuristic so as to minimize the number of interactions with the user. We have created a web-based content creation and editing tool that quickly enables subject matter experts to create and validate PL-bots with minimal training and without requiring a deep understanding of logic or probability. To date, domain experts at a well-known insurance company have successfully created and deployed over 80 PL-bots to help insurance agents determine customer eligibility for policy discounts and endorsements.

1 Introduction

Employees, agents, customers and other interested parties of many businesses often need to determine whether or not a business rule or guideline applies in a given context. For instance, a homeowners insurance policy holder might wish to determine if their new diamond ring is covered in the case of theft, and if not what endorsement they need to add, while an insurance agent might want to learn if a one of their clients qualifies for a certain discount.

Although the customer or employee may have access to web sites, policy contracts or other documentation describing the business rule, such sources are often difficult to find. Moreover, even if documentation is located, the rule may require high cognitive load to understand or be outright incomprehensible by the seeker due to jargon, ambiguity or inherent complexity. This leads to an increased burden on corporate call centers as confused and frustrated customers or employees seek an expert for help.

Conversational interfaces, chatbots, and related intelligent user interfaces are promising emerging technologies that have the potential for addressing many of these concerns (Zamora 2017; Perez and Pascual 2011). A successful intelligent virtual assistant would interact with users to understand their questions or concerns. After it has determined that the user wants to know whether a certain business rule applies, it would do so by asking the user the fewest number of questions. Furthermore, the virtual agent would be intelligent enough to allow the user to communicate in natural language.

This paper reports on efforts we have made toward creating efficient and intelligent conversational interfaces. We present an approach for creating intelligent virtual agents, which we call probabilistic-logic bots (PL-bots), that empower users to quickly evaluate complex business rules by minimizing the number of questions the bot asks of the user. We have deployed PL-bots as part of a conversational agent that aims to assist insurance agents in answering questions about eligibility for insurance coverage discounts and endorsements for new and existing customers. We have also designed and deployed tools that allow domain experts, who in general have non-analytical backgrounds, to quickly create, edit and manage PL-bots with little training.

Our tools allow users to publish stateless APIs to be consumed by stateful chatbot applications. The PL-bots are implemented as look-up functions: the user or chatbot application supplies the ID of the business rule being evaluated and the current *evidence* or set of known facts. The PL-bot either responds with the optimal fact to ask next or the resulting value of the business rule (*i.e.,* true or false) if the supplied evidence is sufficient to evaluate it.

The key technology underlying PL-bots are Bayesian networks (BNs), a member of the class of probabilistic graphical models (see, for example, the Koller and Friedman textbook (Koller and Friedman 2009)). The BNs used by PL-bots encode the logical expressions for the business rule deterministically and the uncertainty over input facts probabilistically.

2 Preliminaries

2.1 Bayesian Networks

A Bayesian network is way of representing the joint probability distribution of a set of random variables that exploits the conditional independence relationships among the

X_1	X_2	$\dots$	X_n	$p(Y=0)$	$p(Y=1)$
0	0	...	0	1.0	0.0
0	0	...	1	1.0	0.0
		...		1.0	0.0
1	1	...	1	0.0	1.0

Table 1: Deterministic CPT for AND node $Y = X_1 \wedge X_2 \wedge \dots X_n$

variables, often greatly reducing the number of parameters needed to represent the full joint probability distribution, while providing a powerful and natural way to represent the dependencies that do exist.

A Bayes net consists of two components: a qualitative one (the structure) in the form of a directed acyclic graph whose nodes correspond to the random variables and a quantitative component consisting of a set of conditional probability distributions. The structure of the graph encodes a set of conditional independence assertions through the absence of arcs between nodes. In particular, a node is conditionally independent of all non-parent ancestors given its parents. These assertions allow the full joint probability distribution to be compactly represented by storing a conditional probability distribution at each node conditioned on its parents, as can be readily seen through a rewriting of the chain rule.

$$\begin{aligned}\Pr(X_1,\dots,X_n) &= \prod_i^n \Pr(X_i|X_1,\dots,X_{i-1}) \\ &= \prod_i^n \Pr(X_i|parents(X_i)\end{aligned}$$

Here the X_i's are the random variables and $parents(X_i)$ is the parent set for X_i. The last line exploits the conditional independence relationships and the ordering from 1 to n is such that a node is preceded by all of its parents.

We use conditional probability tables (CPTs) to represent the conditional distributions (at present PL-bots support only discrete-valued random variables). Rows of the CPT for X give conditional distributions of X given a joint setting (or *configuration*) of X's parents, and the columns correspond to values of X. Nodes whose value is given by a deterministic function of its parents have deterministic CPTs in which each row has a single probability of 1 with the rest of the values in that row equal to 0. We refer to nodes with deterministic CPTs as deterministic nodes.

2.2 Propositional Logic in Bayes nets

It is straightforward to represent propositional logic expressions in Bayesian networks using Boolean variables with deterministic CPTs whose values come from the truth table of the expression. Let $Y, X_1, \dots, X_n$ be Boolean variables where

$$Y = f(X_1, \dots, X_n)$$

is defined by Boolean function $f()$. We can represent Y in a Bayes net by a deterministic node with parents $\{X_1, \dots, X_n\}$ and CPT where $p(Y = 1|X_1, \dots, X_n) = f(X_1, \dots, X_n)$. Table 1, for example, shows a CPT for a conjunction. We use the coding of 0 for false and 1 for true.

A downside of this direct CPT encoding of logical nodes is that since the size required grows exponentially with n; it is not viable for moderately sized n. Fortunately, some important functions including the disjunction and conjunction permit a more compact representation using intermediate variables which we describe in Section 4.1.

3 Probabilistic-Logic Bots

In this section we describe how we construct Bayesian networks to represent business rules and how we use them to drive a conversational interface.

3.1 BNs for PL-bots

A PL-bot Bayes net (PL-BN) has nodes $(F_c \cup F_b \cup F_d \cup C \cup \{T\})$ where F_c, F_b and F_d are sets of nodes for categorical, Boolean and derived facts respectively, C is the set of logical clause nodes and T is the target node.

The categorical facts and Boolean facts (the probabilistic nodes in PL-BNs) constitute the set of variables PL-bots prompt users to provide values for. For this reason we sometimes refer to $F_c \cup F_b$ as *input facts*. For simplicity of exposition we assume all input facts are initially unobserved[1].

Derived facts are binary deterministic nodes that provide one-hot encodings of the categorical nodes where each categorical node is parent of the derived facts for each value in its domain. The clause nodes and the target node are logical nodes which are either a disjunction (OR node) or a conjunction (AND node) of its parents. Each clause node and the target node has two or more binary-valued parents from $F_b \cup F_d \cup C$.

We complete our description of PL-BNs by specifying how we set the conditional distributions. The logical nodes $(C \cup \{T\})$ use the methods for deterministic CPTs described above and in Table 1 (see also Section 4.1 for more efficient representations for nodes with a large number of parents). For probabilistic nodes, which for PL-BNs do not have parents, we only need specify prior probabilities.

One of our design guidelines is to enable users to simply and quickly create PL-bots by employing sensible defaults when possible. We only require designers to specify input facts, clauses and the target rule to create a functioning PL-bot. Input facts are initialized to uniform prior distributions. That this simple approach has yielded surprisingly good results thus far may be explained in part by the fact that a "perfect" PL-BN is one that is correct regarding the next fact to ask about. Perfect probability estimates are not necessary.

However, sometimes it is clear that the default PL-bot solicits facts non-optimally. A common example is when residents of a low-population region are ineligible for a discount. In theory, the business rule could be evaluated with a single query about where the user is located; however, in reality this would rarely result in resolving the rule due to the low population. With uniform priors such a fact has a

[1]It is a straightforward extension to permit PL-bots to begin with non-empty evidence

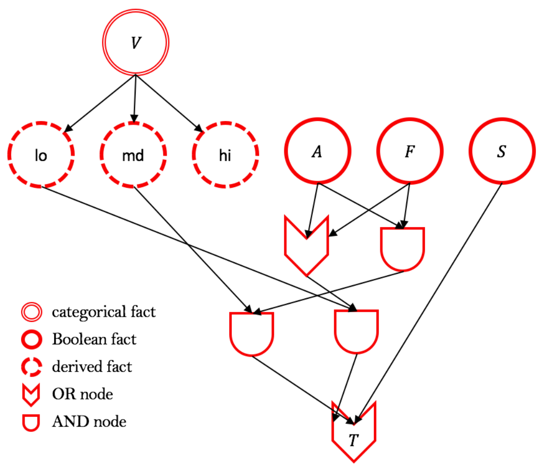

Figure 1: PL-BN structure for the jewelry item endorsement example described in Section 3.1. Endorsement eligibility (target T) depends on facts for the item's value, represented by categorical fact V with values lo, med and hi, whether the home has a burglar alarm (A), whether all gemstones are fixed to a base (F), and whether the jewelry is stored in a permanent locked safe (S).

significantly inflated mutual information with the target, and the default PL-bot will often ask it at the first opportunity. By manually setting the input fact priors, these non-optimal situations can avoided.

Example Consider the following description of a fictional business rule for determining whether an endorsement providing coverage for a jewelry item can be added to a homeowners policy:

> A jewelry item endorsement may be added to a homeowners policy to cover an individual jewelry item if it is stored in a locked safe that resides within the insured building. Otherwise the endorsement is allowed if either of the following hold: i) the item has low value and either the insured building has a burglar alarm or all gemstones are fixed to a stable body, or ii) the item has medium value, the insured building has a burglar alarm and all gemstones are fixed to a stable body.
>
> We can formalize this business rule with

$$T = S \vee ([V = \text{lo}] \wedge (A \wedge F)) \vee ([V = \text{med}] \wedge (A \vee F))$$

where T is the target, V is the value of the item, A indicates if there is a burglar alarm, F indicates if all gemstones are fixed and S indicates if the jewelry item is stored in an appropriate locked safe.

To aid in the implementation and curation of business rules (which sometimes can involve complex Boolean logic), we created an intuitive GUI that allows domain experts to create facts, intermediate rules (which correspond to clause nodes in the BN), and target business rules. Figure 1 shows the resulting PL-BN structure using our tool for the jewelry item endorsement example.

3.2 Conversational Interface

In this section we describe how PL-bots evaluate business rules. Algorithm 1 contains PL-bot pseudocode used after it has determined which business rule the user is interested in evaluating. Discussion of methods for determining the applicable business rule, for example natural language classification of user utterances, are beyond the scope of this paper.

In addition to a Bayes net, each modeled business rule has a set of conversational strings. A PL-bot presents the welcome string at the beginning of a conversation to evaluate that rule. When it asks the user for the value of an input fact it displays the prompt string for the corresponding fact. And when it has determined the value of the rule it displays either the eligible or ineligible string as appropriate.

PL-bots evaluate a business rule by soliciting the values of facts from users, using the (conditional) mutual information between input facts and the target as a heuristic to select the next fact to ask about at each step. The conditional mutual information between X and Y given evidence $\vec{e}$ is

$$I_{\vec{e}}(X,Y) = \sum_{x,y} \Pr(x,y|\vec{e}) \log_2 \left(\frac{\Pr(x,y|\vec{e})}{\Pr(x|\vec{e}) \Pr(y|\vec{e})} \right).$$

$I_{\vec{e}}(X,Y)$ is an information theoretic concept that measures the degree to which learning the value of X when we already know $\vec{e}$ reduces our uncertainty in Y (and vice-versa as it is symmetric).

Algorithm 1: Pseudo-code of PL-bot for evaluating a business rule.

```
function BUSINESSRULEBOT($G, s, \epsilon$)
    $G$: PL-BN for the business rule
    $s$: conversation strings for the business rule
    $\epsilon$: termination threshold

    $T \leftarrow$ target of $G$
        ▷ Initialize unobserved input facts and evidence
    $U \leftarrow$ categorical and Boolean facts of $G$
    $\vec{e} \leftarrow \{\}$

    SAY(s.welcome)
    while $\min_{t\in\{0,1\}} (1 - p(T = t|\vec{e})) > \epsilon$ do
        $F_{next} \leftarrow \arg\max_{F\in U}$ MUTINFO$_{\vec{e}}(F,T)$
        SAY(s.prompt for $F_{next}$)
        $utterance \leftarrow$ USER_INPUT( )
        $value \leftarrow$ EXTRACTVALUE($F_{next}, utterance$)
        append $F_{next} = value$ to $\vec{e}$
        $U \leftarrow U - F_{next}$
    end while
    if $p(T = 1|\vec{e}) > 1 - \epsilon$ then
        SAY(s.eligible)
        return true
    else
        SAY(s.ineligible)
        return false
    end if
end function
```

A PL-bot begins with an empty evidence vector $\vec{e}$. The

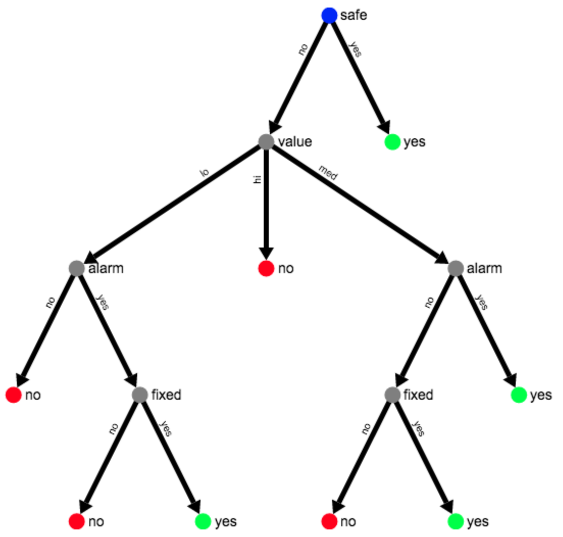

Figure 2: Pre-computed decision tree used to validate model logic and as cache for responsive bot.

categorical and Boolean facts comprise the initial set of unobserved facts. It proceeds by iterating the following steps until the uncertainty in the value of the target falls below ϵ:

1. Calculate the mutual information between each unobserved fact and the target conditioned on the current evidence. The fact with the largest mutual information is the next fact
2. Solicit the value of next fact the from the user, add it to the evidence and remove next fact from the unobserved facts.

4 Practical Matters

In this section we report on our solutions to challenges encountered deploying PL-bots. We focus on general issues rather than those peculiar to our environment

4.1 Sluggish chatbots due to slow inference

The time for determining the next fact should be on the order of at most a few hundred milliseconds. Longer delays lead to sluggish interactions and a poor user experience. The calculation involves a probabilistic inference query to compute $\Pr(F_i, T|\vec{e})$ for each unobserved fact F_i. We have discovered that for many business rules the time needed for this calculation is not acceptable for on-line inference. Therefore, prior to putting a model into production, we pre-compute its decision tree. Figure 2 shows an example of the pre-computed tree for the jewelry example. Besides supporting a more responsive conversation the decision tree is used by bot administrators to visually validate the logic.

However, for some more complicated rules the sheer number of inferences necessary to directly compute the tree make it intractable even in the off-line scenario. It has been long known that exact inference for general BNs is NP-hard in the treewidth of the underlying graph (Cooper 1990). Although efficient inference methods exist for some restricted

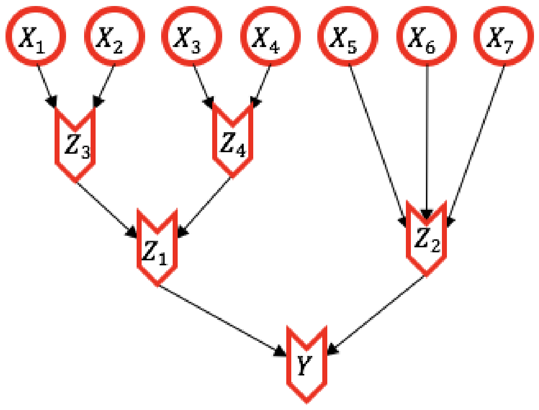

Figure 3: Tree representation for a disjunction of 7 variables $Y = X_1 \vee X_2 \cdots \vee X_7$ using intermediate variables Z_1, Z_2, Z_3 and Z_4 corresponding to the parenthesiziation $Y = [(X_1 \vee X_2) \vee (X_3 \vee X_4)] \vee [X_5 \vee X_6 \vee X_7]$. Representing $\Pr(Y|X_1, \ldots, X_7)$ directly with a CPT requires $2^8 = 256$ values while the tree representation requires only 48 total values in the CPTs of the intermediate nodes and Y. Also, this representation integrates efficiently with exact inference methods such as variable elimination without causing blowup in space or time requirements.

classes of network structures such as chains and trees, PL-BNs is not one of these cases. It is straightforward to show that inference in PL-BNs is NP-hard by reduction to 3-SAT. Thus, we are freed from investigations toward a general polynomial time solutions.

Tree representation of logical nodes Here we report on two heuristic methods that have enabled us to create highly responsive PL-bots without sacrificing exact inference. Conditional distributions of logical nodes $Y = f(X_1, \ldots, X_n)$ for some Boolean functions may be more compactly represented than the 2^n values required by the CPT representations described in Section 2.2[2]. Here we describe a representation of conditional distributions for disjunctions that, through introduction of intermediate variables, requires only $O(n)$ space and integrates efficiently with exact probabilistic inference methods for Bayesian networks. An analogous procedure exists for conjunctions.

Consider the disjunction of k variables:

$$Y = f_{\vee}(X_1, ..., X_k) = X_1 \vee \ldots \vee X_k.$$

Exploiting the associativity of disjunction we introduce intermediate variables Z_1 and Z_2

$$Z_1 = f_{\vee}(X_1, \ldots, X_{k/2})$$
$$Z_2 = f_{\vee}(X_{k/2+1}, \ldots, X_n)$$

and express Y as

$$Y = f_{\vee}(Z_1, Z_2)$$

[2]While it is possible to represent the $\Pr(Y|X_1, \ldots, X_n)$ with only the logical rule and dispense with CPTs entirely, this only reduces the static model size. The operations of inference – when the space demands are greatest – typically realize the tabular representations or reduced forms of it.

X_1	X_2	$p(Y=0)$	$p(Y=1)$	$p(Y=2)$
0	0	1.0	0.0	0.0
0	1	0.0	1.0	0.0
0	2	0.0	0.0	1.0
1	0	0.0	1.0	0.0
1	1	0.0	1.0	0.0
1	2	0.0	1.0	0.0
2	0	0.0	0.0	1.0
2	1	0.0	1.0	0.0
2	2	0.0	0.0	1.0

Table 2: 3-state logic CPT for $Y = X_1 \vee X_2$ where 0, 1 and 2 represent false, true and unsure respectively.

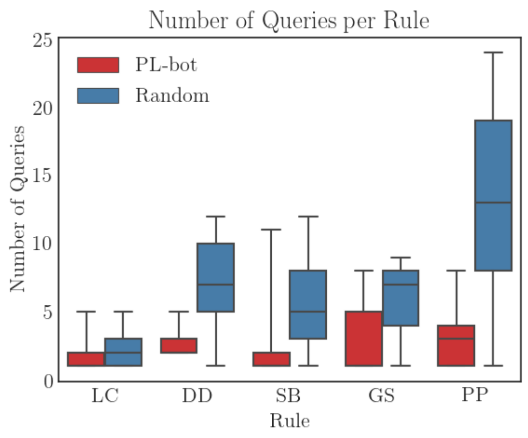

Figure 4: Results from a selection of 5 eligibility rules: livestock coverage (LC), defensive driver (DD), septic backup (SB), good student (GS), and personal property (PP). Here we show the range, median and inner quartiles of the number of queries necessary to evaluate the rule for 1000 simulated users for each querying strategy.

reducing space requirements from $O(2^k)$ to $O(2^{\frac{k}{2}})$. Repeating this process by recursively splitting intermediate variables with more than three inputs yields a tree representation for Y containing $O(n)$ intermediate nodes, each of which (along with Y) have two or three parents. Thus, the aggregate size of all CPTs is also $O(n)$. Figure 3 shows an example of using this process with a disjunction over 7 variables.

Inference by cases In some PL-BNs inference is greatly simplified when a certain conditional fact S, which we call the split node, is known. For these models it may be advantageous to calculate the posterior distribution between unobserved input fact U_i and T $\Pr(U_i, T|\vec{e})$ (needed to compute the mutual information) by marginalizing over S:

$$\Pr(U_i, T|\vec{e}) = \sum_s \Pr(s|\vec{e}) \Pr(U_i, T|s, \vec{e}).$$

The key idea is to trade one complex primary inference query for a number of simpler secondary queries.

We have found this approach to be especially effective for models in which the business rule is a disjunction of

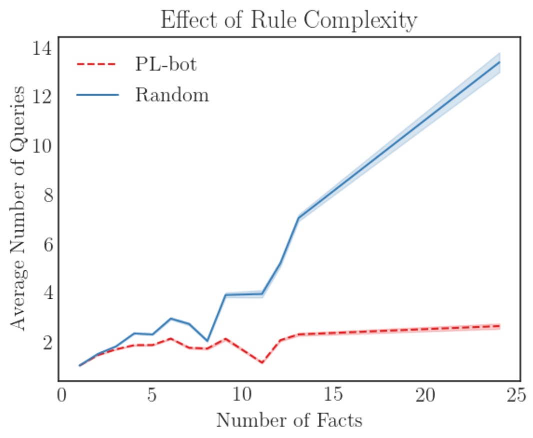

Figure 5: The result of our simulations aggregated by the number of facts involved in the business rule. As expected, the effectiveness of the strategy depends heavily on the complexity of the business rule.

different rules according to the value of a categorical fact that takes a large number of values. For example, because of state-to-state variation in insurance regulatory rules, our organization's defensive driver discount has slightly different eligibility requirements depending on the driver's state of residency. Correspondingly, the PL-bot for this discount has a clause node for each for each state that is true only when the insured driver lives in that state and the other requirements of the rule are met. Each of the state-level clause nodes is a parent of the target node (an OR node).

We have used this strategy successfully with multiple business rules to turn intractable inference problems into ones that run in just a few seconds.

This approach is similar to other conditioning methods for inference in graphical models, such as cutset conditioning (Pearl 1988) and recursive conditioning (Darwiche 2001). These methods search for nodes to condition on (analogous to our split variable) that will yield simpler secondary inference tasks through analysis of the graph. Our approach, on the other hand, uses both the graph structure as well as simplifications due to *determined* nodes that arise not from the structure but the form of the conditional distributions (frequently in deterministic nodes). A determined node is one that while not observed directly is known with certainty given the observed input facts. For example, a conjunctive clause node is determined when any of its inputs is 0. Importantly, the non-probabilistic determined nodes for a given set of evidence can be computed quickly in PL-BNs by cycling through nodes in topological order.

4.2 User uncertainty of input fact values

A user may sometimes be unable to provide the value of an input fact, but there may be other facts that if known allow the business rule to be evaluated. It is straightforward to accommodate uncertain users with a 3-valued logic where Boolean variables take values from {false, true, unsure}. The CPT for 3-state logic extends from its truth table in the obvious way. See Table 2. Note that if any of the parents of an OR or AND node are 3-valued, that node will be 3-valued as well.

5 Experiments

Domain experts used the creation tool described in Section 3.1 to create more than 80 business rules for determining eligibility for discounts or endorsements. The rules comprise a wide range in complexity; some just have a few facts and other more complex rules have more than 20 facts and many intermediate clauses.

For each rule, we randomly created 1000 "users". In this context a user is simply a set of facts drawn from the distribution according to the prior indicated by the creator through our tool. An example user for the the jewelry item endorsement from Section 3.1 might be:

$$\{V = lo, A = True, F = False, S = True\}$$

Which represents a user who has a low value item without fixed gemstones, stored in a safe with a burglar alarm. For each user, we then simulated a "conversation" using both the PL-Bot and random strategies. The strategy chooses a variable to query, the user supplies the value of that variable, and the process stops when the target business rule can be evaluated. For each user and strategy, we record the number of queries necessary to determine the target rule. We show some detailed statistics for 5 individual rules in Figure 4. The personal property (PP) model is the most complex model with 24 facts. Note that the random strategy sometimes had to query the value of every one of these facts, whereas the PL-Bot never had to query more than 10.

Unsurprisingly, we found that the complexity of the rule and number of facts affected the performance of the PL-Bot strategy. This is intuitive; if fewer facts are involved, the random strategy is more likely to select the same variable to be queried as the PL-Bot strategy. Also, the simplest rules with just a few facts can usually be evaluated with 1 or 2 queries either way. However, we see a remarkable difference with the more complex rules. In Figure 5 we plot the average number of queries versus the number of facts for all of our business rules.

6 Conclusion

We have described an approach to building intelligent virtual assistants that help users determine whether or not business rules apply in a given context. Our approach, probabilistic-logic bots (PL-bots), employs Bayesian networks to represent the deterministic and probabilistic aspects of business rules in a single model. The deterministic part comes from the business rule's logic while the probabilistic part comes from the uncertainty as to the value of input facts upon which the rule depends.

A PL-bot evaluates a business rule through interactions with a user where in each interaction the bot asks the user for the value of an unknown input fact, using mutual information as a heuristic to minimize the number of questions asked.

Subject matter experts create PL-bots using a content creation tool that does not require any advanced understanding of Bayes nets, logic or probabilistic models. To date our organization has successfully created and deployed more than 80 PL-bots to assist insurance agents and other internal users.

References

Cooper, G. F. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence* 42(2-3):393–405.

Darwiche, A. 2001. Recursive conditioning. *Artificial Intelligence* 126(1-2):5–41.

Koller, D., and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Perez, D., and Pascual, I. 2011. *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices*. IGI Publishing.

Zamora, J. 2017. Rise of the chatbots: Finding a place for artificial intelligence in India and US. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion*, IUI '17 Companion, 109–112.