Actionable Recommendations for Small Businesses With Hybrid AI

Sudhir Agarwal[1], Lalla Mouatadid[1], Anu Sreepathy[1], Kevin Furbish[1], Morgen Kimbrell[1] and Mike Gabriel[1]

[1]*Intuit Inc., 2700 Coast Ave., Mountain View, CA 94043*

firstname_lastname@intuit.com

Abstract

We present a business use case for computing actionable recommendations for SMBs to prevent near-future or existing critical situations for which a Hybrid AI solution seems more promising than a pure Machine Learning or a pure Symbolic AI solution. For one of the most common of such critical situations, namely, insufficient operating funds, we provide details on the complexity of the problem as well as the requirements on actionable recommendations. We argue why the requirements cannot be satisfied by an end-to-end Machine Learning or Symbolic AI system. We show a breakdown of the monolith into sub-problems, and justify the selection of the most appropriate AI technique for each component, in particular, Machine Learning, Symbolic Rules, and Mathematical Optimization. We also discuss some of the challenges industry would face when adopting Hybrid AI solutions.

1. Introduction

The US is home to over 32 million small and medium businesses (SMBs) and they make up 99.7% of all US employer firms [1]. Due to their nature, SMBs are generally vulnerable to changes in their environment and ecosystem and thus regularly face a lot of challenges. For example, over 80% of US SMBs fail because of cash flow problems. Recently, this became more widely known as SMBs worldwide suffered huge losses due to the COVID-19 pandemic. Providing recommendations to SMBs ahead of time to navigate critical situations can help them thrive in the long term. However, for recommendations to be effective, they must be actionable, that is, relevant, comprehensible, optimal for each SMB's preference criteria, and compliant with respect to SMBs legal context. Intuit®, the global technology platform that makes TurboTax®, QuickBooks®, Mint®, Credit Karma®, and Mailchimp®, helps consumers and small businesses overcome their most important financial challenges. It serves over 5.3 million SMBs through its accounting software QuickBooks[1] [2]. Making such actionable recommendations accessible via our software would thus enhance our SMB customers' experience with our products as well as cultivate their confidence.

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022), Stanford University, Palo Alto, California, USA, March 21–23, 2022.

[1]QuickBooks is an accounting software provided by Intuit that allows small and medium sized businesses (SMBs) to do their own accounting: from issuing invoices and managing inventory to organizing their payrolls and tracking their bills and payments.

We deploy many Machine Learning (ML) models at Intuit, from predicting cash flow to calculating the approval rate of businesses applying for loans. While these and similar ML models perform well for detecting possibly complex situations such as insufficient operating funds in the near-future, end to end ML models fall short on computing actionable recommendations for preventing these critical situations due to low accuracy on relevance for the customer, lack of explainability or guarantee for compliance. On the other hand, a pure Symbolic AI system would require a large knowledge base which is not only expensive but also infeasible for tasks requiring pattern detection for which the rules may not be known.

Traditional recommender systems can be classified as: content-based, collaborative, knowledge-based, demographic and utility-based. Content-based recommenders (CB) rely on the user's previously preferred items in order to recommend similar items [3]. Collaborative filtering-based recommenders (CF) rely on the choices and opinions of other people who share similar interests as the user in order to recommend an item [4]. Knowledge-based recommenders (KB) recommend items based on knowledge about the user, the item, and their relationships [5]. To achieve higher performance and overcome the drawbacks of traditional recommendation techniques, [6] proposed a hybrid technique that exploits the best properties of two or more recommenders, where the most common combinations often include CF techniques in an attempt to avoid cold-starts, sparseness or scalability problems [7, 8].

Unlike traditional product recommender systems, a system that recommends the actions an SMB can take next has to take into consideration a multitude of factors. For example, to avoid having insufficient operating funds, such a system has to identify potential sources of income and savings and categorize them by their availability, probability of success, and cost in case of success. Furthermore, it has to compute an optimal, compliant and explainable combination of income and saving actions while considering their potential impact on SMB's financial burden and relationship with its customers.

In this paper, we present a business use case for computing actionable recommendations for SMBs to help prevent near-future or existing critical situations for which a Hybrid AI solution seems more promising than a pure ML or a pure Symbolic AI solution. We are encouraged by previous comparisons [9] of Hybrid AI to traditional methods used to build "good performing" recommender systems in the financial industry, specifically in the areas of credit evaluation, portfolio management, and financial prediction management. These combinations of ML and Symbolic AI provide a rich set of possible approaches to consider for our focus on small business cash flow. In Section 2, we present the problem of computing actionable recommendations in detail by listing the requirements for such a solution to be practical for our SMB customers. We illustrate the requirements with a common and important case of cash flow problems faced by SMBs leading to insufficient operating funds [10]. The need for a Hybrid AI approach was primarily dictated by the complexity of the problem. We present how we broke down the monolithic problem into components and our analysis of how each component might best be solved in Section 3. On breaking down the problem, we noticed that some of the components are already solved at Intuit. For example, we already have ML models for predicting the likelihood of an SMB's customer paying their invoice on time. In Section 3 we also present how we solve other components with the most appropriate AI technique for them, as well as how various

components are connected to achieve the overall solution [2]. Overall, our solution uses a mix of ML models, Symbolic Inference, and Mathematical Optimization. We conclude in Section 4 by summarizing our contribution, present adoption challenges of Hybrid AI solutions in industry and possible next steps.

2. Use Case

Each one of our SMB customers, hereon referred to as our users, is in a unique business and at a different stage. Hence, we cannot build accurate solutions that generalize across different users. Our approach consists of identifying complex events that our users might face; breaking them down into small manageable events, and coming up with personalized, targeted, step-by-step solutions. These are the actionable recommendations we aim to provide. Different steps in the process of computing actionable recommendations will need different techniques, ranging from ML models to symbolic rules and mathematical optimization. The final solution must be realistic, achievable, explainable and transparent.

Formally, given a problem measured by some cost function, we want to present our users with a solution, or combination thereof, that minimizes the cost. The desired features of these solutions must be *explainable*, e.g., we chose such and such solution because it increases x or decreases y, *transparent*, e.g., we believe such a solution has a success probability of x, *actionable*, e.g., following the proposed steps in the solution, you can minimize your cost by x, and here is a plan to do so, and *realistic*, e.g., we recommend this solution because of the following relevant actions to you, etc.

1. Preventing Near-Future Insufficient Operating Funds

To make the description of our approach more concrete, let's consider the example of how to prevent near-future insufficient operating funds. We assume that the prediction of a critical situation, such as insufficient operating funds, has already taken place, and focus on computing actionable recommendations for SMBs to prevent or effectively handle such a critical situation.

Since this is a user based, and not an "others-like-you" based, recommendation system that also needs to meet compliance requirements, it would require incorporating expert knowledge into the solution. In particular, one can think about how financial experts would present such solutions: they would gather all the options available to their customer, evaluate each one based on the customer's unique situation, and tailor their advice to their customer. To achieve similar levels of personalization, we first break down the overall problem of computing actionable recommendations to sub-problems, choose the right AI technique for each sub-problem, and orchestrate their respective solutions to achieve the overall solution.

Assuming such a system exists, this would allow us to present our customers with *actionable, explainable recommendations*. A recommendation from this system can be as verbose as:

> "We predict that you will run out of operating funds in n days. We recommend delaying paying your vendor w since you have a grace period of m days. We also recommend

[2] The solution presented is a proof of concept that has neither been tested nor deployed at scale. While most components have been fully implemented, some are only partially complete.

applying for a loan of $\$x$ since you have been pre-approved by y for this amount. We also recommend asking for the payment of invoices from z customers because they usually pay on time/they often use your discounts/they have exceeded their payment terms".

To produce such recommendations we need to understand as many of the constraints and preferences as possible that each user has. For instance, to recommend a loan in the example above, we need to be able to evaluate the user's approval for different loans. To match a user to a lender, we need to understand the constraints of the lender, e.g., they only want to lend to businesses that have been profitable for at least 6 months. Being able to include these constraints and preferences into our system allows us to explain our solution and better approximate the success likelihood of our solution.

These recommendations are not only actionable to reduce the described cash flow problems, but are also explainable by nature. The solution can include the success likelihood computed for each step, and we can go a step further to allow users to modify our proposed solution since they know their customers better. We want such a system to meet the following requirements:

- *Accuracy*: We want our recommendations to be compliant and the success likelihood and penalty cost calculations to be as accurate as possible.
- *Speed*: Especially when optimizing the combination that includes the user's preferences.
- *Configurability*: To allow users to weigh in on proposed solutions.
- *Explainability*: As we described above.
- *Data Efficiency*: This is a user-specific recommendation system. Unlike other systems that learn from the behavior of all users, this system needs to be tailored to learn from individual users.

3. Computing Actionable Recommendations with Hybrid AI

Assuming that the initial step, the near future forecasting problem, has already been predicted, our use case reduces down to the following steps: 1. Identifying potential remedial actions. 2. Computing the success probabilities and cost of each action in Step 1. 3. Selecting relevant actions. 4. Generating plans using optimal combinations of solutions from Step 3.

There are different ways an SMB owner can try to prevent running out of operating funds. Some we cannot predict—getting financial help from a family member, for instance. Others we can predict, for example being approved for a loan. These actions are what we call *remedial actions*. Once these actions have been identified in Step 1, and considering each SMB's constraints, such as its contracts with vendors, the next step is to calculate the cost associated with each action. For example, getting a new loan could impact the credit score of the business. This impact is the associated cost of this action. For every remedial action, we also calculate its success probability. For instance, loan A is guaranteed from QuickBooks Capital, but loan B is pre-approved with certain conditions and thus has a lower success probability. We refine remedial actions further by taking into account the SMB's preferences. For instance, an SMB

owner might be against taking on additional debt. The final step is to optimize for low cost and success probability of action plans to present to the user.

As one can see, the components of such a system cannot all rely on just pure ML or pure symbolic reasoning. A hybrid AI approach is fundamental to designing such a solution, We need a system that can make near future predictions (ML), identify potential solutions (Symbolic), calculate the success likelihood of each component of the proposed solution (ML), select relevant actions (Symbolic), and finally optimize for the best combination (Mathematical Optimization). Figure 1 below illustrates the different building blocks of our recommendation system. Let's dive into each step in detail next.

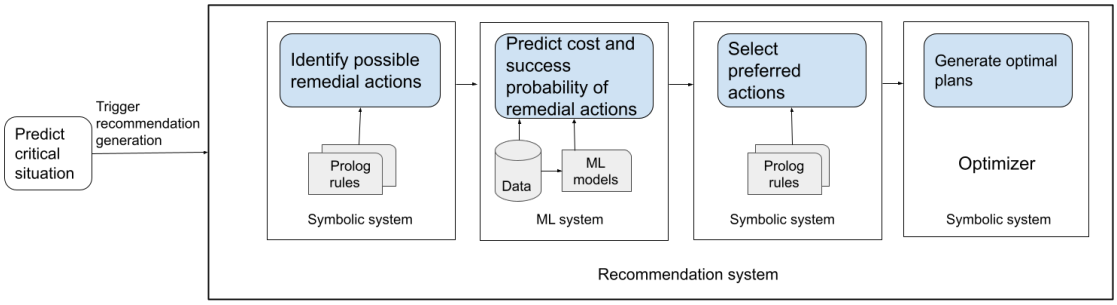

Figure 1: Overview of our approach to building an actionable recommender system.

1. Identify Possible Remedial Actions

In this step, we identify the set of actions that can help prevent the critical situation. Each SMB can be in a unique situation depending on its size, insurances, contracts, and other legal obligations. Given such possibly complex logical constraints, an ML model is not likely to perform well for this task even if we had a lot of training data. In addition, expert knowledge in the form of heuristics for this task is more easily available and incorporating it can add value. Therefore, we chose a rule based approach for this step. We use SWI-Prolog [11] for rule based inference via the SWI-Prolog Python wrapper PySwip [12].

For our example of insufficient operating funds, some possible remedial actions are: (i) soliciting customers to pay early (ii) applying for loan, and (iii) delaying bill payment. We first define the following corresponding relations: (a) *customer* with arguments *customerId, amount, cost*, and *successProbability* (b) *bill* with arguments *billId, type, amount, cost*, and *isDelayed* (c) *loan* with one argument *amount*.

We can now define the following rules for selecting possible actions. Since these criteria are not implemented as code but externalized as declarative rules, the criteria can easily be modified by developers, a user at the SMB, or a financial advisor.

- $isPossible(bill(C, D, E, F, G))$:- $employee(smb, A), length(A, B), B =< 20,$ $hasBill(smb, C), bill(C, D, E, F, G)$. This rule selects all bills of the given SMB smb if it has at most 20 employees.

- $isPossible(customer(A, B, C, D) :- hasCustomer(smb, A), \backslash+ isPremiumMember(A)$. This rule selects all customers that do not have a premium membership contract.

2. Compute the Success Probabilities and Costs of Possible Actions

Computing success probabilities and costs of possible actions depends largely on user data and hence we choose to implement this using ML models. For predicting the success probability of a solicited customer, the features we use include: (i) the amount due and its due date, (ii) the payment history of said customer, (iii) the percentage of the last k invoices that were paid late, as well as the percentage of invoices paid late, and (iv) the average amount paid on time by the customer. A model that specifically predicts whether and when an open invoice will be paid on time has already been implemented at Intuit. We adapt this model to calculate the success probability for a solicited customer using the features above. We also need to take into account the probability that a customer would agree to an early payment if presented with a discount or some other perks offered by the SMB.

For predicting the probability of qualifying for a loan, we use the following features to model credit worthiness: (i) Business bank statements, (ii) Income tax returns, (iii) Balance sheets, and (iv) Operating history. For bill deferral, the success probability is 100% since it is the decision of the user.

For calculating the cost of requesting an early payment from each customer, we use: (i) the length of the relationship between the customer and the business, (ii) the number of transactions between the customer and the business and (iii) the ratio of revenue from the customer to the total business revenue. For learning the costs associated with bill deferrals and loan approvals, we use: (i) the impact on the user's credit score, (ii) the interest fees and (iii) the late fees associated with deferrals. For now however, we have handcrafted these costs for the purpose of our proof-of-concept (POC) system. The next step is to learn them using the features above. Finally, we normalize these costs across all remedial actions and compute a total weighted cost for each recommended plan. The weights (by default all equal 1) model user preferences for the suggested remedial actions.

3. Select Relevant Actions

An SMB may wish to exclude some relevant actions from further consideration for several reasons, such as the SMB's preferences for certain vendors, current situation regarding delayed bills, pending loans, and delayed loan instalment payments, etc. False positives may lead to undesirable plans. On the other hand, it is not only more accurate but also easier and faster to encode the rules to infer relevant actions as Symbolic AI rules just as a financial consultant would systematically deduce relevant actions based on expertise and experience. Therefore, we chose a rules based approach to solve this step.

For the purpose of computing relevant actions, we use the *isRelevant* query predicate that can be defined with the help of Prolog rules, for example as follows. Only relevant bills and customers obtained by evaluating $isRelevant(X)$ are passed on to the next step of computing optimal plans.

- $isRelevant(bill(X, T, A, C, 0))$:- $isPossible(bill(X, T, A, C, 0)), A =< 1500$. This rule selects all bills that are not delayed and whose amount is at most 1500.
- $isRelevant(bill(X, \text{“}Utility\text{”} A, C, 0))$:- $isPossible(bill(X, \text{“}Utility\text{”} A, C, 0))$, $A =< 5000$. This rule selects all utility bills that are not delayed and whose amount is at most 5000.
- $isRelevant(customer(X, A, C, P))$:- $isPossible(customer(X, A, C, P))$, $A =< 5000$. This rule selects all customers that owe at most 5000.
- $isRelevant(customer(X, A, C, P))$:- $isPossible(customer(X, A, C, P)), P >= 0.9$. This rule selects all customers that will pay with a probability higher than 0.9.

4. Compute Optimal Plans

Recommended plans should be optimal, explainable, and tailored to the SMB. But the data required to build an ML model for computing such plans is not easily available. A mathematical optimizer seems to be the easiest and best choice for this step. For the POC implementation we use the Python implementation of the Simplicial Homology Global Optimization (SHGO) [13] from the SciPy package.

For a given insufficient operating funds situation and a set of relevant actions for a given SMB, where each relevant action has an amount, a cost, and a success probability, we first compute two optimal plans. Either of them can cover the required cash flow but one of them minimizes the cost, while the other maximizes the success probability (minimizes failure probability).

Variables: A plan essentially computes a vector of $k + l + 1$ amounts, where k is the number of customers that may pay early, l is the number of bills whose payment may be delayed, and the last element for loan.

Bounds: Bounds is also a vector of length $k + l + 1$ to hold the lower and upper bounds for each item in the variable vector. The first k bounds are $(0, customer(i, amount))$ where $customer(i, amount)$ denotes the amount due on i^{th} of the k customers. The next l bounds are $(0, bill(i, amount))$ where $(0, bill(i, amount))$ denotes the amount of the i^{th} of the l bills to be paid by the SMB. The last bound is $(0, -cashflow)$ to denote that a loan higher than what is needed to avoid running out of operating funds should not be considered.

Constraints: There is only one constraint that the sum of all the amounts in the plan must be equal to the predicted amount of inadequate operating funds.

Objective Function: For the cost minimization plan, the objective function is the sum of all costs for the respective amounts in the variables vector. We compute such costs assuming linear dependency of cost on the amount as $cost = \sum_{i=1}^{k} \frac{plan(i,amount) \times customer(i,cost)}{customer(i,amount)} + \sum_{i=k+1}^{k+l} \frac{plan(i,amount) \times bill(i-k,cost)}{bill(i-k,amount)} + loancost(loan\_amount)$. Similarly, for the success probability plan, the objective function is the average failure probability (note that the optimizer tries to minimize the objective function) for the amounts in the variables vector. The failure probability is simply *1 - success probability*. The failure probability of the variable vector is the average of all non-zero amounts in the vector.

Having the minimum cost plan and the maximum success probability plan, we compute the rate of change of success probability with increasing cost assuming linear dependency for

simplicity. This allows us to compute optimal plans for the given success probability or the given cost that, for example, can be entered by the user after reviewing the first two plans. We compute such a fixed cost or fixed success probability plan by adding to the above problem specification a constraint to specify the fixed value.

4. Conclusion, Next Steps, and Challenges

In this paper, we presented an industry use case for offering actionable recommendations to SMBs to help resolve a near-term insufficient operating funds. We argued why a pure ML based or a pure symbolic approach is not the best way to solve the problem and why a hybrid approach might be the way out. We presented a Hybrid AI approach to solve the problem, detailing the subsystems of the recommendation system and reasoning for choosing an ML or a symbolic technique for each subsystem.

While the solution presented takes into consideration three broad options, namely, requesting customers to pay early, obtaining a loan and delaying payment of bills, and finds an optimal combination of the options to alleviate the problem, one can imagine including other possible options such as relocating the business to a newer location with a cheaper rent, cutting down certain operational costs, etc. The proposed recommendation system can be expanded to include other types of recommendations of the type "grow your business".

Ideally, this system can be built as a configurable system that takes in a set of rules depicting possible actions, corresponding ML models to predict success probabilities and cost of each action, a set of constraints and a cost function. An orchestration layer runs the rules in a rule engine to filter the possible set of actions that can then be used by the optimizer to generate one or more actionable plans for the intended type of recommendation.

The same sequence of techniques might not apply to all SMBs; there might not be sufficient data available to predict success probabilities of actions for a business that has just gotten started. For such businesses, a symbolic rules based approach might be the only option to use to begin with, while an ML based solution becomes an option as more data gets collected. All of these scenarios raise the following questions that the team would like to work on as next steps: What is the boundary where one switches from a rules based approach to a learning approach? How does one make the two approaches complement each other? What is the process of reconciliation when they contradict each other?

In general, implementing such a solution at scale in the industry comes with its own set of challenges. There are no mature tools, frameworks or best practices that make implementation of a Hybrid AI system straightforward. In addition, organizational structures with teams consisting solely of ML experts might lack the skills to build a rule based system and vice versa. Given a succession of models, where one's output feeds into the next, or an ensemble of models that are used as an aggregate, there is also no standard way to compute the accuracy across an entire system and this is an interesting area to explore in itself.

Evaluating the performance of such a system is also a challenge as there are no existing solutions solving this problem to use as a comparison. We expect to use a qualitative approach to evaluate the efficacy and value of this system with a cohort of early stage test users. Surveys and interviews with members of the test cohort would reveal how often these preliminary

users employ a solution generated by the above described recommendation system, and the value of the recommended solutions to the user. Intuit has a partner network of internal and external financial advisors and accountants who could also contribute to the evaluation process by indicating if they agree with the system generated advice for the unique circumstances facing a cohort user.

References

[1] U. S. B. A. O. of Advocacy, US small business statistics, 2021. URL: https://cdn.advocacy.sba.gov/wp-content/uploads/2021/11/03093005/Small-Business-FAQ-2021.pdf.

[2] Intuit®, Intuit-investor-day-2021-presentation, 2021. URL: https://s23.q4cdn.com/935127502/files/doc_presentations/2021/Intuit-Investor-Day-2021-Presentation.pdf.

[3] M. J. Pazzani, D. Billsus, Content-based recommendation systems, in: The Adaptive Web: Methods and Strategies of Web Personalization. Volume 432 of Lecture Notes on Computer Science, Springer-Verlag, 2007, pp. 325–341.

[4] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems 22 (2004) 143–177.

[5] R. Burke, Hybrid recommender systems: Survey and experiments, User Modeling and User-Adapted Interaction 12 (2002) 331–370. doi:http://dx.doi.org/10.1023/A:1021240730564.

[6] R. Burke, Hybrid Web Recommender Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 377–408. URL: https://doi.org/10.1007/978-3-540-72079-9_12. doi:10.1007/978-3-540-72079-9_12.

[7] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 734–749. doi:10.1109/TKDE.2005.99.

[8] A. Bellogín, I. Cantador, F. Díez, P. Castells, E. Chavarriaga, An empirical comparison of social, collaborative filtering, and hybrid recommenders, ACM Trans. Intell. Syst. Technol. 4 (2013) 14:1–14:29. URL: https://doi.org/10.1145/2414425.2414439. doi:10.1145/2414425.2414439.

[9] A. Bahrammirzaee, A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems, Neural Computing and Applications 19 (2010) 1165–1195.

[10] D. Higley, New report reveals cash flow struggles of small businesses and self-employed, 2018. URL: https://quickbooks.intuit.com/r/cash-flow/cash-flow-small-business-self-employed/.

[11] J. Wielemaker, T. Schrijvers, M. Triska, T. Lager, Swi-prolog, Theory and Practice of Logic Programming 12 (2012) 67–96. doi:10.1017/S1471068411000494.

[12] Y. Tekol, other contributors, Pyswip. a python - swi-prolog bridge to enable querying swi-prolog in python programs, 2017. URL: https://github.com/yuce/pyswip.

[13] S. C. E. Endres, C. Sandrock, W. W. Focke, A simplicial homology algorithm for lipschitz optimisation, Journal of Global Optimization 72 (2018) 1573–2916. doi:10.1007/s10898-018-0645-y.