# Multi-Stage Prompting for Next Best Agent Recommendations in Adaptive Workflows

**Prerna Agarwal[1], Harshit Dave[2*], Jayachandu Bandlamudi[1], Renuka Sindhgatta[1], Kushal Mukherjee[1]**

[1]IBM Research AI
[2]ABV-IIITM Gwalior India

## Abstract

Traditional business processes such as loan processing, order processing, or procurement have a series of steps that are pre-defined at design time and executed by enterprise systems. Recent advancements in new-age businesses, however, focus on having adaptive and ad-hoc processes by stitching together a set of functions or steps enabled through autonomous agents. Further, to enable business users execute a flexible set of steps, there have been works on providing a conversational interface to interact and execute automation. Often, it is necessary to guide the user through the set of possible steps in the process (or workflow). Existing work on recommending the next agent to run relies on historical data. However, with changing workflows and new automation constantly getting added, it is important to provide recommendations without historical data. Additionally, hand-crafted recommendation rules do not scale. The adaptive workflow being a combination of structured and unstructured information, makes it harder to mine. Hence, in this work, we leverage Large Language Models (LLMs) to combine process knowledge with the meta-data of agents to discover NBAs specifically at cold-start. We propose a multi-stage approach that uses existing process knowledge and agent meta-data information to prompt LLM and recommend meaningful next best agent (NBA) based on user utterances.

## Introduction

Traditional business applications rely on prescribed steps that are prescriptive, codified, and automated using enterprise resource planning (ERP) applications or workflow engines. New-age businesses, however, rely on agile and flexible ways of executing tasks. The flexibility is not about pre-defining all possible ways of running a process, such as a hospital admission process or a claim handling process (Margaria et al. 2012). It is instead about enabling the business users to freely execute the process tasks with applicable business rules and well-defined goals. Both the traditional and the adaptive approaches are goal-driven as they aim to meet a business goal. But, adaptive process does not have a detailed prescription of how the set of process steps are to be executed, but only prescribe what has to be achieved, which are translated as specific goals by process owners (insurance claim needs to be processed, hospital patient needs to be discharged, and so on). The adaptive process allows business users a certain degree of freedom to stitch a sequence of process tasks to achieve their goals. These process tasks have been going through the modernization phase via the use of Application Programming Interfaces (APIs). For example, Salesforce, as a leading enterprise customer relationship management (CRM) provider, offers APIs to enable CRM capabilities into a large number of custom systems (Vukovic et al. 2016). Additionally, as automation-enabling technologies evolve, conversational interfaces have been explored to seamlessly allow business users to perform their tasks (Chakraborti et al. 2022). Unlike task-oriented dialog systems that deal with slot filling to perform an action for an intent (Rastogi et al. 2020), chat interfaces for executing process tasks require calling an ordered sequence of APIs. These APIs are referred to as agents (a common terminology used in automation (Chakraborti et al. 2022)). Further, a single agent can be associated with multiple process tasks. For example, the process task for updating an order would require two agents $\{api\_get\_order, api\_save\_order\}$, while the task of approving an order would require agents $\{api\_get\_order, api\_approve\_order, api\_notify\}$. Hence, the next best agent (NBA) recommendation varies in the context of the process task.

Previous works on recommending the NBA when executing a process (Rama-Maneiro et al. 2023) differ from our current approach as they: i) focus on traditional business processes having predefined sequences of process tasks, and ii) use historical execution data from structured information and, iii) do not have conversational utterances of the user. The focus of our approach is a scenario where an enterprise has an existing process, such as their human resource procurement, or order processing supported by a set of agents (or API[1]). Additionally, the users converse in natural language to execute process tasks. Our approach utilizes the meta-data information available in the agent and the process knowledge to recommend the next best agent by using a Large Language Model (LLM). We address the cold-start problem as new agents continually get added with business

*The work was carried out during internship at IBM Research


[1]we use agent for API from here on

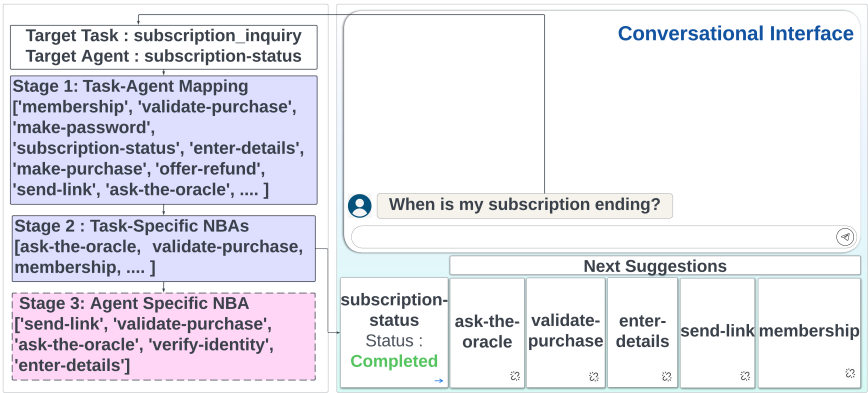



Figure 1: NBA Recommendations in Conversational UI

process steps continually getting automated and improved.

Using LLMs in enterprise applications comes with challenges. Most real-world applications require some customisation of the knowledge in the LLM (fine-tuning), which requires data and can get outdated. Prompting has shown great promise for a personalised, controlled output. Given the limited prompt size of different LLM models, it is not possible to describe the full set of agents in a single context. Hence, to support a collection of potentially hundreds (or thousands) of changing agents, a staged approach is required to narrow down the search for the right set of agents.

An illustration of our proposed staged approach is shown in Figure 1. As shown, based on the user utterance, the *target agent* and *target task* for which the NBA needs to be recommended is identified. In the first stage, a set of agents possibly required to accomplish the identified process *target task* are identified. In the next stage, the top-$k$ task-oriented NBAs are obtained out of the agents obtained from the first stage. In case, the user-utterance does not contain the *target task* information, the global set of top-$k$ NBAs across all process tasks are identified. Hence, the main contributions of our paper are as follows:

1. A process-aware prompting approach using LLMs for Next Best Agent (NBA) Recommendations in Adaptive Workflows at cold-start based on user utterance.
2. A multi-stage approach architecture to infuse the process knowledge with meta-data information of agents in a process task-oriented manner.
3. A demonstration of the approach providing meaningful NBAs using only the meta-data information surpassing different baselines.

## Background Terminologies

We provide the core terminologies used in our paper:

- ***Agent:*** refers to an atomic function corresponding to an operation of the API. When an agent is invoked in a conversational interface, it receives the required input parameters, performs the operation and returns the output. An agent definition includes meta-data such as agent description, input-output signatures, API call type (GET, POST, etc.), and additional details required to execute it.
- ***Process Task:*** refers to the high-level task (or activity) of a business process. Usually, multiple agents are sequenced together to accomplish a process task. For e.g.: consider a process task *Interview Scheduling* for a candidate of a Recruitment Process. To accomplish this process task, the sequence of agents to be executed are as follows: *map_panel_with_candidate*, *send_slots_to_candidate*, *confirm_slot*. Note that, each agent can be used in multiple process tasks.
- ***Process Knowledge:*** Business process knowledge comprises the process task descriptions, persona information authorized to perform certain process tasks and business rules such as *A job requisition must be approved before receiving applications*. This knowledge often changes over time.

## Proposed NBA System Framework

The overall system framework is shown in Figure 2. First, the agent specification and the process task specification are used to derive a mapping between the agents implementing a specific task. As new agents get added, their specification is used to update the information of the tasks they can automate. At run time, the user interacts with the conversational interface by providing a natural language utterance to accomplish a process task. The *target task* and the *target agent* can be identified from the user utterance with the help of task and intent identification models based on the process and the agent information available in the agent database and process repository. As there exists mature work in intent and slot filling, we assume this is done by existing tools[2]. Once the *target agent* and *target task* are identified, the task and agent mapping is used to provide top-$k$ NBA recommendations that can possibly be sequenced next. Here again, with the process knowledge, agent meta-data information, and the specific task and related agents, the LLM gets instructed by a prompt at different stages of the system. There may be scenarios where the user utterance only maps to a *target agent*. In such a scenario, all the tasks associated with the *target agent* are identified and the NBAs in the context of multiple relevant process tasks are computed. The prompt at each stage depicts the objective and a few in-context input-output example pairs. Our architecture follows a reasoning chain i.e., the output of one stage goes to the next one leading to-

[2]https://www.ibm.com/products/watson-assistant

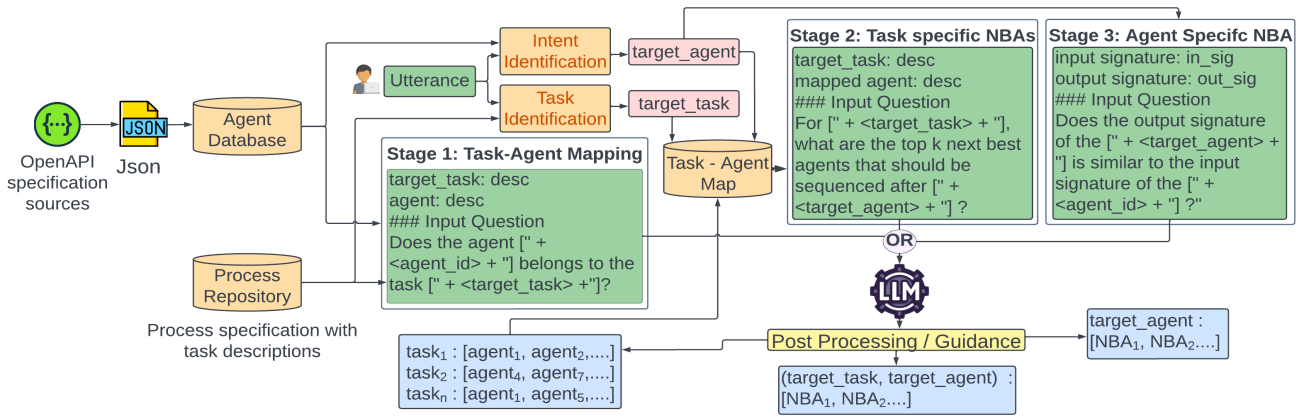



Figure 2: Next Best Agent (NBA) Recommendation System Framework using LLMs

wards obtaining NBA recommendations. We now provide details of each stage.

## Offline (Stage 1): Task to Agent Mapping

The agent database can contain hundreds (or thousands) of agents to support a flexible and adaptive workflow. Due to the limited prompt size of LLMs, it is not possible to describe all the agents in the prompt. Hence, the objective of this stage is to identify a small set of agents $|A_t| << |A|$ where $A_t$ are the set of agents that can possibly be used to accomplish the *target task* $t$ and $|A|$ is the total number of agents $A$ in the database. As shown in Figure 2, given a pair of *target task* $t$, an agent $a \in A$ and their descriptions, the prompt is designed to ask the model if agent $a$ maps to the *target task* $t$ or not. The task to agent mapping is done for all tasks $t \in T$ where $T$ is the set of all tasks. The name and description of the task and agent make the LLM aware of their objective and hence help to identify their alignment. We provide two in-context examples with input as process task and agent descriptions and output as yes/no. Figure 1 shows the sample output of this stage. The task and agent mapping is stored and used for identifying NBA.

## Online (Stage 2): Task Specific NBAs

The task to agent mapping could be a many-to-many relationship i.e., one task can have many agents, and an agent can be used in many tasks based on the context. First, the *target agent* and the *target task* are identified based on the user utterance. Next, the NBAs of the *target agent* corresponding to the *target task* are obtained. As shown in Figure 2, given the description of *target task*, description of all agents $A_t$ obtained from *task and agent mapping*, the prompt is designed to ask the model for the top-$k$ next best agents that can be sequenced after the *target agent*. We provide two in-context examples with output as the list of top-$k$ NBAs. Hence, this stage provides the top-$k$ NBA agents of the *target agent* w.r.t *target task* $t$. Figure 1 shows the sample output at this stage for a given user utterance. In case, the target task is not identified from the user utterance, the top-$k$ NBAs of the *target agent* are obtained w.r.t all tasks $t \in T_a$ where $T_a$ is the set of tasks where the *target agent* $a$.

## Online (Stage 3): Agent Specific NBA

This stage is executed when the *target task* is not obtained from the user utterance. In this scenario, the objective of this stage is to obtain top-$k$ NBAs among the top-$k$ NBAs obtained for all tasks $t \in T_a$ obtained for the *target agent*. This is because when the agents are sequenced to accomplish a process task, the data flows from one agent to the next one. Therefore, in this stage, we take the top-$k$ agents for all tasks $t \in T_a$ and rank them based on whether the output of the *target agent* $a$ semantically matches with the input of the NBA agent to obtain agent-specific top-$k$ NBA agents. As shown in Figure 2, given a pair of *target agent* $a_t$, an agent $a \in NBA_{agents}$ obtained from the previous stage, their descriptions and their input-output signatures, the prompt is designed to ask the model if the output of agent *target agent* $a_t$ is similar to the input of the agent $a$ or not. We provide two in-context examples with answer in yes/no.

This stage is not required if the *target task* is present in the user utterance. This is because, in adaptive workflows, at run-time, a user may jump to a new agent altogether without worrying about the data flow (similar to web page surfing). Hence, determining the ranking between the task-specific top-$k$ NBAs is not required. However, such ranking becomes essential to obtain the global top-$k$ NBAs from a bunch of top-$k$ task-specific NBAs.

## Grounding LLM Output

LLMs can hallucinate while generating the output, and hence grounding becomes extremely important. Grounding is the process of using LLMs with relevant information to enable domain-specific and accurate responses. We employ 2 different techniques to ground the output: (1) Guidance based controlled generation (2) Post-processing.

**Guidance based Controlled Generation:** We use *guidance*[3] language by Microsoft for controlled text generation in LLM. This language can be used with decoder-only models. A simple syntax is used to structure the output with multiple generations, selections, conditionals, etc. The output guidance rule used at each stage is shown below:

[3]https://github.com/microsoft/guidance

**Stage 1** `Output:` {`select "answer" options= options`}. This rule selects the answer among the options provided i.e., yes/no. for this stage.

**Stage 2** `Output :` {∼ `gen "answer" max_toke ns=200 stop="\n"`}. This rule generates the answer maximum upto 200 tokens until \n is not encountered. It follows the generation observed in in-context examples and performs controlled generation.

**Stage 3** `Output :` {`select "answer" logprob s='logprobs' options = options`}. This rule selects the answer among the options provided i.e., yes/no for this stage, along with the log-probability. The log-probability then helps to choose the top-$k$ agents whose log-probability for a class "yes" is the highest.

**Post-Processing:** To ground the answers generated by encoder-decoder models, we process the output using BERT.

**Stage 1 & Stage 3** We perform basic preprocessing on the generated answer i.e., remove punctuation, extra spaces, and check whether the text contains yes or no, ignoring the letter cases to ground the response.

**Stage 2** We ground the generated answers to agent database based on agent similarity calculated using Sentence Transformers[4] `all-MiniLM-L6-v2` model as follows: (1) the extra text apart from the agent name is filtered out (2) each agent name is validated against the agent database. For the agents whose names do not exist in the agent database, we replace it with the most similar agent (if similarity exceeds 0.95) otherwise, that agent is discarded. Additionally, if the number of agents mapped to a process task (*target task*) exceeds the prompt token limit of the LLM, we calculate the maximum number of agents $m$ that can be added in the prompt as follows: We compute the average number of tokens required by (1) agent description (2) process task description (3) two in-context examples (4) question. We then calculate the log probability of each (task, agent) pair mapping for both *yes* and *no* class (in stage 1), and pick the top-$m$ agents whose mapping log-probability for *yes* class is the highest.

# Experimental Setup

## Baselines

To the best of our knowledge, this is the first system to address NBA recommendations at cold-start. Hence, there are no publicly available baselines present. Therefore, we create a baseline *NBA using BERT* that uses meta-data information of agents. Additionally, we use two baselines that are information retrieval-based search indices to find relevant top-$k$ agents. Search baselines are evaluated in two steps: 1) The search index is created using all descriptions from agents' meta-data 2) The search index is queried to identify top-$k$ relevant agents using the *target_agent* description. We use two popular choices for the search i.e; Keyword based search using Elastic search (Gormley and Tong 2015) and Semantic search using ChromaDB[5] as the baselines.

[4]https://www.sbert.net/

[5]https://github.com/chroma-core/chroma

• **NBA using BERT:** For this baseline, we follow a 2-step approach: (1) We assign the agents to each process task based on the cosine similarity calculated between the sentence embeddings of their respective descriptions, (2) For a given *target_agent* and *target_task*, we then compute the cosine similarity and match the inputs-output signatures between the agents assigned to *target_task* in step 1 and the *target_agent* to provide top-$k$ NBAs using the sentence embeddings from `deepset/sentence_bert` model.

• **ChromaDB based semantic search**: ChromaDB is a vector DB that supports semantic search on documents. For this baseline, we first obtain the sentence embedding for all agent descriptions using Sentence Transformers `all-MiniLM-L6-v2` model and create a search index using the sentence embedding as features. We retrieve top-$k$ similar agents ranked on the cosine similarity of the embedding of *target_agent* and all agents.

• **Elastic Search based keyword search**: Elastic search (Gormley and Tong 2015) supports Keyword based search. In this baseline, we first obtain features using BM25(Robertson and Zaragoza 2009) method for all the agent descriptions, and the search index is created using the BM25 features. This search index is queried using a similar approach as above for semantic search.

## Datasets

There exists no open-source dataset of agents with ground truth NBA recommendations. Hence, we experiment with one open-source dataset and transform it to evaluate NBA. Additionally, we use an IBM internal dataset for evaluation:

• **Action Based Conversational Dataset (ABCD)**(Chen et al. 2021): This dataset provides goal-oriented dialogue data with sequences of actions to achieve a task. We leverage only the meta-data of tasks and actions. We further transform each action in the format of the agent manually. The number of agents and tasks in this dataset are 30 and 10, respectively.

• **Recruitment Process**(RP): This is an IBM proprietary customer dataset that consists of process tasks and agents to accomplish various tasks of the recruitment process, such as opening of job positions, shortlisting candidates, interview onboarding, etc. The number of agents and tasks in this dataset are 43 and 24, respectively.

## Models

Among the set of open-source LLM models, we choose the following models for our experimentation:

• **Flan-T5-XXL-11B** (Chung et al. 2022): It is an encoder-decoder model from Google. It is fine-tuned with instructions for better zero-shot and few-shot performance. The prompt limit of this model is 512 tokens.

• **OpenLlaMA-13B** (Touvron et al. 2023): It is one of the finest open-sourced decoder-only model from Meta with prompt limit of 2048 tokens.

• **MPT-7B** (Team 2023): It is a decoder-only model from MosaicML that use a modified transformer architecture optimized for efficient training and inference. The prompt limit of this model is 2048 tokens.

| Models | Dataset | Avg. R % | Avg. P % |
|---|---|---|---|
| Flan-T5 | ABCD | 0.739 | 0.481 |
| | RP | 0.695 | 0.548 |
| Open LlaMA | ABCD | 0.908 | 0.431 |
| | RP | 0.84 | 0.288 |
| MPT | ABCD | **0.912** | 0.3875 |
| | RP | 0.86 | 0.316 |
| HE | ABCD | 0.289 | 0.6245 |

(a) Stage 1 - Task to Agent Mapping

| Models | Agent List | Dataset | NBA Acc.% | Agent R % |
|---|---|---|---|---|
| Flan-T5 | Stage 1 | ABCD | **0.5125** | **0.844** |
| | | RP | 0.393 | 0.556 |
| | Ground Truth | ABCD | 0.6875 | 0.844 |
| | | RP | 0.369 | 0.611 |
| | HE | ABCD | 0.1 | 0.111 |
| Open LlaMA | Stage 1 | ABCD | 0.475 | 0.6 |
| | | RP | **0.464** | **0.694** |
| | Ground Truth | ABCD | 0.4875 | 0.6 |
| | | RP | 0.357 | 0.583 |
| | HE | ABCD | 0.1 | 0.133 |
| MPT | Stage 1 | ABCD | **0.5125** | 0.622 |
| | | RP | **0.464** | 0.583 |
| | Ground Truth | ABCD | 0.6875 | 0.778 |
| | | RP | 0.524 | 0.667 |
| | HE | ABCD | 0.1125 | 0.133 |

(b) Stage 2 - Task specific NBAs

| Models | Dataset | NBA Acc.% | Agent R% |
|---|---|---|---|
| Flan-T5 | ABCD | 0.212 | 0.411 |
| | RP | 0.286 | 0.418 |
| Open LlaMA | ABCD | 0.287 | 0.4705 |
| | RP | 0.321 | 0.226 |
| MPT | ABCD | **0.288** | **0.705** |
| | RP | **0.345** | **0.488** |

(c) Stage 3 - Input Output Flow Matching (Optional)

Table 1: Results of each stage of proposed NBA recommendation framework

| Baselines | Dataset | NBA Acc.% | Agent R% |
|---|---|---|---|
| NBA using BERT | ABCD | 0.183 | 0.269 |
| | Recruitment | 0.25 | 0.417 |
| ChromaDB | ABCD | 0.243 | 0.384 |
| | Recruitment | 0.369 | 0.583 |
| Elastic Search | ABCD | 0.097 | 0.269 |
| | Recruitment | 0.4047 | 0.583 |
| Our Framework | ABCD | **0.5125** | **0.844** |
| | Recruitment | **0.464** | **0.694** |

Table 2: Comparison with baselines

## Evaluation Metrics

**Stage 1** We report precision (P) and recall (R) to evaluate this stage. However, the aim of this stage is to provide a small subset of agents to choose NBAs; hence, we use recall as our primary metric.

**Stage 2 & 3** We define 2 evaluation metrics here: (1) *NBA Accuracy*: the fraction of total number of NBAs predicted correctly out of the total number of NBAs present in the Ground Truth (GT) across all agents (2) *Agent Recall*: the fraction of number of agents for which atleast 1 NBA was predicted correctly in top-$k$. Our goal is to provide atleast 1 NBA correctly to each user request, hence, we use Agent Recall as our primary metric for these stage(s).

## Results

The evaluation results of each stage of our proposed framework (for $k = 5$) on both ABCD and Recruitment datasets are shown in Table 1. The ABCD dataset is designed for goal-oriented dialogues, therefore, we also ask the Human Evaluators (HE) to manually label task to agent mappings and task-specific NBAs for each agent given the meta-data and process knowledge. We compare this manual labelling (denoted by HE) with the ground truth provided in the dataset and report the metrics in Table 1a and Table 1b.

As shown in Table 1a, the average recall of task to agent mapping for MPT is the highest (0.912, 0.86), followed by OpenLlaMA (0.908, 0.84) and then Flan-T5-XXL (0.739, 0.695) for both ABCD and Recruitment dataset respectively. We observe precision v/s recall trade-off, and hence, we see the reverse order of performance for average precision. As expected, the average precision of HE is the highest, whereas the average recall is the lowest. To re-iterate, we focus on average recall as the primary metric for this stage.

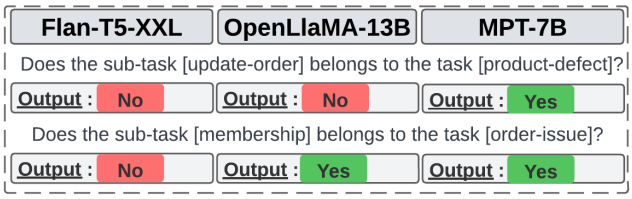
For Stage 2, we perform **ablation study** using different sources of task to agent mappings (denoted by Agent List) as shown in Table 1b, i.e. Agent List obtained from: (1) Stage 1 (2) Ground Truth provided in the dataset (3) HEs.

• *Stage 1*: ABCD Dataset: NBA Accuracy and Agent Recall % are the highest for MPT. This is followed by Flan-T5-XXL and OpenLlaMA. RP Dataset: NBA accuracy is higher for OpenLlaMA and MPT followed by Flan-T5-XXL.

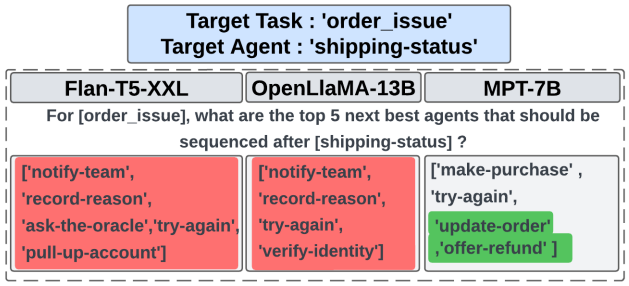
• *Ground Truth*: ABCD Dataset: both NBA Accuracy % and Agent Recall % is the highest for MPT and Flan-T5-XXL followed by OpenLlaMA. RP dataset: Accuracy is highest for MPT followed by Flan-T5-XXL and OpenLlaMA for the Recruitment dataset. We observe only a marginal improvement in the metrics as compared to the mappings obtained from Stage 1 which denotes that Stage 1 mappings are more or less complete.

• *HE*: NBA Accuracy and Agent Recall % is the highest for MPT followed by OpenLlaMA and Flan-T5-XXL for ABCD dataset. The overall performance with mappings obtained from HE is very less as compared to the mappings from Stage 1 and Ground Truth. This could be because HEs tend to mark the task to agent mappings only where they are absolutely sure. This also results in low recall at Stage 1 and justifies why we choose high recall as the primary metric.

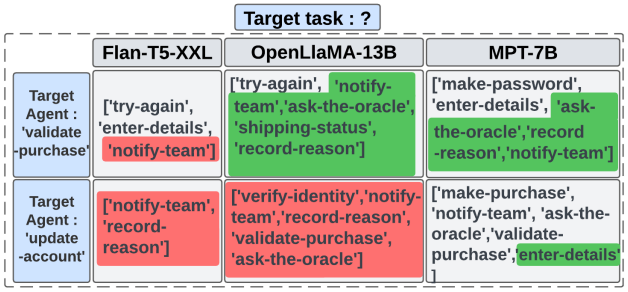
For Stage 3, as shown in Table 1c, both NBA Accuracy % and Agent Recall % are the highest for MPT, followed by OpenLlaMA and Flan-T5-XXL. Overall, NBA Accuracy and Agent Recall % decreases as compared to Stage 2 because the NBAs obtained in this stage are global i.e., across all tasks. Figure 3 shows the qualitative comparison of the LLM models. Overall, we obtained the best results using MPT followed by OpenLlaMA and Flan-T5-XXL model for all three stages. Figure 3a, 3b, and 3c shows a sample example where we demonstrate the output obtained from all 3




(a) Stage 1




(b) Stage 2




(c) Stage 3

Figure 3: Comparison of LLM Models on different Stages of the pipeline (Best viewed in color)

models, keeping the input prompt to be the same. The output highlighted in red indicates the wrong answer, and the ones highlighted in green indicates the correct answer.

**Baseline Comparison:** Table 2 shows the comparison of our proposed framework with the baselines. For both datasets, our framework significantly outperforms all the baselines. Among the baselines, ChromaDB performed the best. For the Recruitment dataset, we observe Elastic Search baseline performs the best because the agent descriptions in this dataset contain high keyword overlap.

## Deployment and Impact

We implement our system in *python* using the *Langchain*[6] framework. Langchain enables the use of any available LLM. Further, helps in our staged pipeline of propagating the output from one stage to the other. We use the models available on Huggingface and perform model inference on A100 GPUs. As shown in Figure 2, we expect the agents to follow standard OpenAPI specification format with standard 3.0 version compliant meta-data fields. The current pipeline is in the process of integration with IBM Watson Orchestrate[7] offering. A sample interface of how NBA recommendations are given to the user is shown in Figure 1. Till date, the product has over 600 APIs wrapped up as agents. We have evaluated our approach on the Recruitment process, however, the product offers support for several business agents with a conversational interface enabled. These services include Asana, Box, Oracle HC, Salesforce, ServiceNow, SAP SuccessFactors, Jira, Slack, Gmail, etc. We aim to provide seamless agent sequencing by providing meaningful NBAs at even cold-start that most of the systems do not provide. Even though the NBAs may not be highly accurate, the expectation is to provide user-relevant recommendation to invoke NBA. Choosing the right NBA among these hundreds of agents is otherwise a manual and time-consuming effort for a user. Hence, even a modest improvement in NBAs would be significant.

## Related Work

Existing work on business process monitoring has recommendations of next best action recommendation models that learn from historical logs (Branchi et al. 2022; Weinzierl, Dunzer et al. 2020). These works, however, are applicable only to static workflows and not to adaptive workflows where there can be multiple ways of executing a task. Recent work (Yaeli et al. 2022) provides the next best agent recommendation approach to support more flexible and ad-hoc workflows. However, this work relies on the historical usage of agents and further provides recommendations at cold-start using deterministic rules. These rules can get outdated for large or rapidly evolving workflows.

Recent developments in LLMs have shown promising results on various downstream tasks, including tools usage, and API calls. With the recently introduced Toolformer (Schick et al. 2023), GPT-4 (OpenAI 2023) and Gorilla (Patil et al. 2023), the importance of using LLMs for invoking API calls has been recognized, encouraging studies in employing API calls. Some of these studies augment the pre-trained LLMs by prompting, and others instruction fine-tune the model. However, existing work does not focus on recommending the next best API (or agent) leading to a user accomplishing a task or a goal.

## Conclusion and Future Work

In this paper, we propose a first of a kind framework to provide next best agent (NBA) recommendations using meta-data and process knowledge for cold-start using Large Language Models (LLMs). Given the limited prompt size of LLMs, we present a staged approach that prompts the LLM at different stages and provides both process task-oriented NBAs and global NBAs based on user utterance. We show that our proposed approach significantly outperforms the baselines. We believe that the recommendations can be further improved with few-shot historical data points as and when available. Further, we plan to collect user feedback for current NBA recommendations and improve prompt engineering and hence, the pipeline.

[6]https://python.langchain.com/

[7]https://www.ibm.com/products/watson-orchestrate

## References


Branchi, S.; Di Francescomarino, C.; Ghidini, C.; Massimo, D.; Ricci, F.; and Ronzani, M. 2022. Learning to Act: A Reinforcement Learning Approach to Recommend the Best Next Activities. In Di Ciccio, C.; Dijkman, R.; del Río Ortega, A.; and Rinderle-Ma, S., eds., *Business Process Management Forum*, 137–154. Cham: Springer International Publishing. ISBN 978-3-031-16171-1.

Chakraborti, T.; Rizk, Y.; Isahagian, V.; Aksar, B.; and Fuggitti, F. 2022. From Natural Language to Workflows: Towards Emergent Intelligence in Robotic Process Automation. In *Business Process Management: Blockchain, RPA, and and CEE Forum, 2022, Proceedings*, volume 459 of *LNBIP*, 123–137.

Chen, D.; Chen, H.; Yang, Y.; Lin, A.; and Yu, Z. 2021. Action-Based Conversations Dataset: A Corpus for Building More In-Depth Task-Oriented Dialogue Systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, 3002–3017. Online: Association for Computational Linguistics.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; Webson, A.; Gu, S. S.; Dai, Z.; Suzgun, M.; Chen, X.; Chowdhery, A.; Castro-Ros, A.; Pellat, M.; Robinson, K.; Valter, D.; Narang, S.; Mishra, G.; Yu, A.; Zhao, V.; Huang, Y.; Dai, A.; Yu, H.; Petrov, S.; Chi, E. H.; Dean, J.; Devlin, J.; Roberts, A.; Zhou, D.; Le, Q. V.; and Wei, J. 2022. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416.

Gormley, C.; and Tong, Z. 2015. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 1st edition. ISBN 1449358543.

Margaria, T.; Boßelmann, S.; Doedt, M.; Floyd, B. D.; and Steffen, B. 2012. Customer-Oriented Business Process Management: Vision and Obstacles. In *Conquering Complexity*, 407–429. Springer.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2023. Gorilla: Large Language Model Connected with Massive APIs. arXiv:2305.15334.

Rama-Maneiro, E.; et al. 2023. Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. *IEEE Trans. Serv. Comput.*, 16(1): 739–756.

Rastogi, A.; Zang, X.; Sunkara, S.; Gupta, R.; and Khaitan, P. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8689–8696.

Robertson, S.; and Zaragoza, H. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3: 333–389.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761.

Team, M. N. 2023. Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs. Accessed: 2023-05-05.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Vukovic, M.; Laredo, J.; Muthusamy, V.; et al. 2016. Riding and thriving on the API hype cycle. *Commun. ACM*, 59(3): 35–37.

Weinzierl, S.; Dunzer, S.; et al. 2020. Prescriptive business process monitoring for recommending next best actions. In *International Conference on Business Process Management*, 193–209. Springer.

Yaeli, A.; Shlomov, S.; Oved, A.; Zeltyn, S.; and Mashkif, N. 2022. Recommending Next Best Skill in Conversational Robotic Process Automation. In Marrella, A.; Matulevičius, R.; Gabryelczyk, R.; Axmann, B.; Bosilj Vukšić, V.; Gaaloul, W.; Indihar Štemberger, M.; Kő, A.; and Lu, Q., eds., *Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum*, 215–230. Cham: Springer International Publishing. ISBN 978-3-031-16168-1.