

# Adaptive Orchestration of Modular Generative Information Access Systems

Mohanna Hoveyda  
Radboud University  
Nijmegen, The Netherlands  
mohanna.hoveyda@ru.nl

Harrie Oosterhuis  
Radboud University  
Nijmegen, The Netherlands  
harrie.oosterhuis@ru.nl

Arjen P. de Vries  
Radboud University  
Nijmegen, The Netherlands  
arjen.devries@ru.nl

Maarten de Rijke  
University of Amsterdam  
Amsterdam, The Netherlands  
m.derijke@uva.nl

Faegheh Hasibi  
Radboud University  
Nijmegen, The Netherlands  
faegheh.hasibi@ru.nl

## Abstract

Advancements in large language models (LLMs) have driven the emergence of complex new systems to provide access to information, that we will collectively refer to as modular generative information access (GenIA) systems. They integrate a broad and evolving range of specialized components, including LLMs, retrieval models, and a heterogeneous set of sources and tools. While modularity offers flexibility, it also raises critical challenges: How can we systematically characterize the space of possible modules and their interactions? How can we automate and optimize interactions among these heterogeneous components? And, how do we enable this modular system to dynamically adapt to varying user query requirements and evolving module capabilities?

In this perspective paper, we argue that the architecture of future modular generative information access systems will not just assemble powerful components, but enable a self-organizing system through real-time adaptive orchestration – where components' interactions are dynamically configured for each user input, maximizing information relevance while minimizing computational overhead. We give provisional answers to the questions raised above with a roadmap that depicts the key principles and methods for designing such an adaptive modular system. We identify pressing challenges, and propose avenues for addressing them in the years ahead. This perspective urges the IR community to rethink modular system designs for developing adaptive, self-optimizing, and future-ready architectures that evolve alongside their rapidly advancing underlying technologies.

## CCS Concepts

• **Information systems** → **Information retrieval**; **Search engine architectures and scalability**.

## Keywords

Adaptive information systems, Graph-based orchestration

## ACM Reference Format:

Mohanna Hoveyda, Harrie Oosterhuis, Arjen P. de Vries, Maarten de Rijke, and Faegheh Hasibi. 2025. Adaptive Orchestration of Modular Generative Information Access Systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*, July 13–18, 2025, Padua, Italy. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3726302.3730351>

## 1 Introduction

Search engines have been the dominant gateway to the vast amount of information online for decades by now. However, traditional search systems are increasingly inadequate as the user's information demands evolve [3, 87, 88]. Today's users no longer settle for a static list of result documents to be manually sifted through; they expect direct answers to questions that might require complex reasoning [47, 74], a synthesis of different pieces or modalities of information [76, 91], or even carrying out tasks or transactions on behalf of them – all delivered through a seamless interactive multimodal interface [3].

Recent breakthroughs in the development of large language models (LLMs) have provided highly promising advancements in multi-turn dialogue capabilities, improved intent detection, and richer content synthesis. These advancements open new opportunities for developing future information retrieval (IR) systems and conversational assistants, and can help tackle some of the persistent challenges in information access in the era of generative AI [3, 87]. We will refer to these future systems as *Generative Information Access (GenIA) systems*, to emphasize our focus on the long-standing problem of information access while acknowledging the importance of generative AI to achieve this objective.

While future GenIA systems could be monolithic LLMs, we see many advantages for a new system architecture, consisting of ensembles of diverse components, with clearly identifiable roles, that interact with each other in order to generate answers to user queries [38, 43, 46, 86, 95].<sup>1</sup> E.g., retrieval can employ a variety of models, including sparse, dense, and graph-based models, each tailored for particular tasks or corpus types [34, 44, 56, 61]. Post-retrieval modules further refine the information being passed to the language model, leveraging natural language inference filters [83], atomic-fact extractors, or even controlled noise injection [15] to



This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGIR '25, Padua, Italy

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1592-1/2025/07

<https://doi.org/10.1145/3726302.3730351>

<sup>1</sup>We purposely write *identifiable* instead of *identified*, since our perspective posits that the roles of components should be learned and adaptive.

ensure more robust answer formulation. For complex queries, such as those requiring multi-hop reasoning [79] or logical inference [74], approaches such as neuro-symbolic pipelines expand the space of possible modules with symbolic solvers [23, 52]. When queries demand task execution, a variety of tools can be incorporated [84]. Likewise, domain-specific tasks may be addressed using specialized or variably scaled LLMs [8, 82].

Taken together, these components and implementations highlight a shift towards ensembles of specialized, interoperable modules that collectively enable effective information access or conversational assistance. A variety of frameworks [12, 24, 32, 40, 45, 96] has been proposed to chain specialized modules into end-to-end systems. However, these solutions often require manually designed pipelines [12, 32] or rely on a static *orchestration* of modules [96].

In this perspective paper, we posit that modularity in GenIA systems introduces a highly important novel research direction into our field: the development of dynamic system designs for information retrieval systems that automatically adapt the modules in its pipeline and the interaction between them. Focusing research on learning the optimal interactions between modules is crucial to advancing modern, generative information access systems. We propose that an adaptive orchestration framework should dynamically coordinate module selection and sequencing based on the properties of the query, the capabilities of the modules, and computational constraints. Thereby, such frameworks will not merely improve response quality but balance quality and accuracy with latency and other costs. Orchestration frameworks should learn from real-time feedback – i.e., through user satisfaction scores, error detection, or runtime metrics – such that the orchestration strategy can be continuously refined, over time. Crucially, this adaptability also paves the way for incremental integration of new modules without necessitating a complete overhaul of existing system infrastructure, a highly desirable property from the view of operations. Central to this vision are three key questions:

- (1) *How can we rigorously define the space of possible modules?*
- (2) *How can we model, automate, and optimize interactions among these heterogeneous components?*
- (3) *How should this modular system dynamically adapt to varying user query requirements and evolving module capabilities?*

We offer provisional answers to these questions through the proposal of a novel framework that addresses these challenges. Our framework relies on a graph-based representation [96], where nodes and edges correspond to modules that represent the subtasks of a pipeline, what agents/tools perform these tasks and the usage of information resources. To achieve adaptive orchestration of systems planning and reinforcement learning algorithms can be used to find and construct the most effective pipeline-design graph for each incoming user request. Thereby, a balance between effectiveness and efficiency is optimized.

After discussing related work in Section 2, our perspective on the future of information retrieval systems is detailed in Section 3. We propose a general framework by which our perspective can be realized in Section 4, which also lays out different implementation choices. In Section 5 we provide an example instantiation of our proposed framework for an adaptive question answering system

that uses a Contextual Multi-Armed Bandit (CMAB) for optimization [41]. Section 5.4 provides experimental results that evaluate this instantiation and reveal it successfully adapts to balance efficiency and effectiveness. Finally, we discuss our perspective and proposed approach in the concluding Section 6.

## 2 Background

### 2.1 Multi-agentic LLM orchestration

Several studies have proposed frameworks for enabling multiple language model-based agents and related modules to communicate to solve tasks [1, 26, 77]. Most work focuses on orchestration without further optimization of the structure of these agents [77, 95]. Inspired by Minsky’s *society of minds* (SoM) [51], which describes how smaller parts of a system can collaborate to achieve a goal, Zhuge et al. [95] suggest a shift from relying on optimizing a single model for solving a task to the optimization of information flow between two or more models. Zhuge et al. [96] propose GPTSwarm to optimize a society of language model-based agents. GPTSwarm structures a system made of multiple agents as a graph, and every computational operation (e.g., querying a large language model with a prompt) is represented as a node within that graph. The dual-level optimization approach of GPTSwarm optimizes prompts at the node level while also enhancing the flow of information by pruning out edges and nodes that are not found useful. While innovative, GPTSwarm proposes an optimization framework that only yields a single static final graph which fails to adapt to different user needs and queries and the varied capabilities of its underlying components.

### 2.2 Mixture of experts (MoE)

A paradigm that is related to, but different from the society of mind (SoM) framework is the *mixture of experts* (MoE) approach [37, 66]. MoE architectures typically partition model capacity across multiple *expert* networks, each specialized in handling specific parts of the input space. A gating or routing network then determines which expert(s) to activate for a given input, facilitating more efficient model scaling and potentially better task performance. Recent advances in MoE have focused on improving routing strategies to reduce computational overhead, balance the load among experts, and refine expert specialization [30, 93].

While both use modular components and routing, MoE learns expert specialization implicitly through training and uses internal vector-based communication, whereas in SoM, modules are explicitly designed for predefined functions with natural language communication enabling higher interpretability. Additionally, SoM allows for heterogeneous modules with diverse capabilities, unlike MoE’s typically homogeneous expert structure.

### 2.3 Adaptivity in IR systems

The need for adaptivity in IR systems has received increasing attention in recent research, with varying interpretations and objectives across different contexts. Parry et al. [57] introduce the notion of adaptively selecting the number of in-context-learning (ICL) examples for queries with varying levels of complexity. Deng et al. [19] advocate for a pro-active conversational agent that can adaptively decide when to intervene and engage the user. In the domain of

retrieval-augmented generation, when to retrieve and when to rely on an LLM’s parametric knowledge based on a given query is a challenging and interesting problem [31, 48, 65]. Mallen et al. [48] propose a binary threshold-based framework that distinguishes whether questions contain popular or long-tail entities, and decides whether to retrieve accordingly. Following this work, several other approaches have been proposed to tackle this problem using an LLM as a classifier [27] or guided by a reinforcement learning-based reward model [6, 67].

### 3 Perspective: Designs of Future GenIA Systems Should be Dynamic, Modular and Adaptive

For a long time, information retrieval systems have relied on static architectures: monolithic search engines, predefined pipelines, and rigid workflows that process user queries in a pre-determined manner. While recent advancements in LLM technologies have enabled the integration of diverse models and tools within these pipelines, the underlying interaction between these components remains largely inflexible, lacking adaptivity and failing to dynamically adjust based on the specific requirements of each input. As the field continues to advance with the introduction of novel and sophisticated models and tools, we believe the time has come for a new architecture of information access systems, where different modules communicate with each other interactively to construct customized information system pipelines, while adapting their interactions based on user input for high throughput, efficiency, and effectiveness.

The need for such adaptability is already evident. User inputs vary widely in their complexity – some require simple fact retrieval, while others demand logical reasoning, disambiguation, or multi-step synthesis. Certain queries may involve task execution, such as booking a flight, while others demand specialized capabilities, like generating code or synthesizing insights from multiple research papers. Yet, most current systems treat all inputs uniformly, applying a static pipeline using the same computational resources, failing to differentiate between trivial and intricate queries. This one-size-fits-all approach is inefficient and suboptimal, leading to unnecessary computational overhead for simple tasks and inadequate depth or poorly suited strategies for complex ones.

What we need is a new architecture for information access systems that does not simply retrieve information but orchestrates an evolving ensemble of specialized models and tools, each playing its role based on its relevance and utility in the moment. A user submits a query, and the system determines which modules to invoke, how they should communicate, and how to optimize their collective output; similar to assembling a team of experts on the fly.

Achieving this vision introduces several challenges. First, we need a framework that allows modules to interact dynamically rather than through fixed orchestration rules. This requires a shift towards expressive *graph-based interaction models*, where modules form transient communication patterns tailored to each query. Second, *adaptivity* must be built into the system itself, such that it learns when to engage certain components, how to weight their contributions, and when to optimize for speed versus depth of reasoning. This demands an optimization mechanism capable of

adjustment in real-time, ensuring that computation is allocated efficiently and based on the query context. Third, as modular systems continue to evolve, new components will emerge, and existing ones will improve. A truly future-ready Information Access system must not only adapt to individual queries but also accommodate continuous changes in its own architecture. This requires mechanisms for dynamically integrating new modules, updating orchestration strategies, and ensuring that the system remains robust as its components evolve. Without such adaptability, the benefits of modularity remain an unrealized promise, a collection of powerful components constrained by a rigid system structure.

The architecture of a GenIA system should go beyond merely assembling a few sophisticated modules; instead, it needs a self-optimizing, adaptive orchestration framework that allows various types of computational processes (next token prediction, retrieval, classification, symbolic reasoning, etc.) to interact meaningfully, in real-time.

In the next section, we propose our framework, to pave the way for the realization of this vision.

## 4 Adaptive Module Orchestration Framework

This section introduces our proposed architecture for modular *GenIA* systems that dynamically adapts to user queries. Section 4.1 describes our main design principles and goals. In Section 4.2 we propose a categorization of modules that make up system designs. Lastly, Section 4.3 discusses how system designs can be dynamically constructed to adapt to individual user queries.

### 4.1 System design principles and goals

The main goal of our framework is to produce an information retrieval system that optimizes a typical balance of effectiveness and efficiency. In other words, we aim to provide a system that is effective at providing high quality responses to user queries, while not incurring too much cost (typically in terms of response time or computational costs). Accordingly, the design of our system should be chosen to optimize our goal, which gives rise to three key questions:






- *What tasks should be executed and in what order?*
- *What tools and agents should be used to execute each task?*
- *What sources of information should be consulted?*

Together, the answers to these questions describe the inner pipeline of a system, and thus, they capture its behavior and cost to operate.

Today’s paradigm to information retrieval system design is to answer these questions once and create a static pipeline that performs the same procedure for every user query. Conversely, we propose a radically different approach and argue that the answers to these design questions should be reconsidered for every user query. Thereby, the system’s design is re-constructed on-the-fly and can adapt to pursue the best effectiveness-efficiency trade-off for individual user queries.

Our dynamic approach means that systems are no longer designed as rigid pipelines, instead, the task of the practitioner is now to develop a space of possible designs that can be searched, with designs that can be quickly constructed and executed. Our framework

**Table 1: Overview of the different module types and their subcategories in our framework.**

Type	Subcategory	Description	Visualization
Tasks	Standalone	Simple operations that do not require interactions between multiple agents and tools.	
	Complex	Multi-step operations requiring agents or interactions between multiple modules.	
Executor	Agents	General execution units (i.e., foundation models) that are widely applicable and can use tools.	
	Tools	Specialized execution units designed for specific tasks with limited applicability.	
Resources (Properties)		Sources of data/information; categorized on the following properties: <i>Structure</i> , <i>Modality</i> and <i>Accessibility</i> .	

enables this through a modular approach, that describes the building blocks that can comprise pipelines; and an adaptive procedure to find, learn and construct pipelines in response to queries.

## 4.2 Modules: Tasks, tools, agents and resources

Since our framework is designed for systems of the future, we aim to encapsulate all existing systems and all of their popular extensions. In order to be future-proof, we apply generic concepts and categories that should remain relevant to future advancements. This also applies to our modular approach, as it only defines on three types modules, each with their own subcategories. Our module types are conceptualized to match the three key design questions posed in Section 4.1: (i) *tasks* (what is done), (ii) *executors* (what does it), and (iii) *resources* (where information comes from). Table 1 provides an overview of our module categories; the remainder of this section describes each category separately. Table 2 provides a list of examples per module type.

### 4.2.1 Task modules

The first type of modules: tasks ( $\mathcal{T}$ ), represent the procedures that can be performed inside a system pipeline. These modules thus describe *what the pipeline does*, but do *not* indicate what performs the procedures or what resources are used. Instead, tasks can have requirements, i.e., which types of executors can perform them, or what resources are needed to complete them. We propose two subcategories for this module type: *standalone* and *complex* tasks.

**Standalone tasks** refer to simple operations that can be executed independently without requiring interactions between multiple executors or complex agents. In general, these are tasks for which specialized tools exist that can only perform that single task. Examples include but are not limited to tasks of *retrieval*, *intent classification*, *query rewriting*, and *direct generation*. Due to their self-contained nature, standalone tasks can be efficiently executed with minimal dependencies, making them well-suited for constructing system designs that are lightweight and fast.

**Complex tasks** refer to more complex operations that can only be executed by multiple executors or complex agents or procedures. In contrast with standalone tasks, these tasks generally involve multiple interdependent operations that cannot be performed by specialized tools. This often means a single action is insufficient to produce a meaningful result, necessitating multi-step reasoning or iterative refinement. Examples include *Retrieval Augmented Generation (RAG)* [38], *Interleaving Retrieval with Chain-of-Thought Reasoning (IRCOT)* [69], and *Logical Inference via Neurosymbolic Computation (LINC)* [52]. In general, complex tasks are less efficient

but more effective, and thus, more appropriate when standalone tasks are ineffective.

### 4.2.2 Executor modules

As their name implies, the executor type of modules ( $\mathcal{E}$ ) can perform tasks within our framework. They serve as the operative entities that process the inputs of a task and decide how to produce outputs that meet the task requirements. Generally, the combination of tasks and executors determine most of the computational costs of a pipeline. We propose two categories of executors: *agents* and *tools*.

**Agents** are general executors capable of performing a large variety of tasks. Unlike tools, agents are not limited to a few well-defined tasks but instead provide flexibility in application, can reason across multiple inputs and may be capable of utilizing tools when performing tasks. The archetype of this category are *foundation models* (especially *LLMs*) acting as generalist problem solvers.

**Tools** are specialized executors designed to perform specific well-defined tasks with a clear operational scope. Thus, tools are characterized by their limited applicability due to their specialization, but this often also results in a great efficiency-effectiveness trade-off. Examples include *classic retrieval models*, *query expansion techniques*, *symbolic solvers*, and various *APIs*. Tools can constitute efficient pipelines for scenarios that are not too complex.

### 4.2.3 Resource modules

Lastly, resource modules ( $\mathcal{R}$ ) define the data sources available to executors. Generally, resource modules indicate what information is used by a pipeline. The choice of resources can affect the computational costs incurred, as larger resources may require more time to process, but also other types of costs if a resource is not freely available. Instead of non-overlapping categories, we categorize by the following properties:

**Unstructured vs. (semi-)structured.** Whether a resource is structured or not is important to what kind of operations can be performed to it. Structured resources (e.g., *relational databases*, *knowledge graphs*, and *taxonomies*) follow predefined schemata, enabling efficient retrieval, precise querying, and logical inference, which benefit symbolic reasoning and fact verification. In contrast, unstructured resources (e.g., *raw text corpora*, *web documents*, *conversational logs*) lack this structure, making such operations more challenging. However, they are often more abundant and may be the only available resource for certain tasks.

**Modality.** The modality of a resource is also important for what tasks and user queries it is useful for. Since a user can request a



**Table 2: Non-exhaustive overview of potential choices for the modules per type (possible nodes in the graphs).**

Node type	Description	
Tasks	<i>Standalone</i>	
	Generation	Generate text (or image and other modalities) given a certain input [10, 17, 68]
	Retrieval	Retrieve pieces of information given a certain input [34, 44, 61]
	Query rewriting	Reformulate/expand queries to improve retrieval performance [21, 92]
	Intent recognition	Categorize user input into predefined intents or discover novel intents [5, 94]
	Asking questions	Asking clarifying questions proactively to engage the user to help the retrieval process [2, 85]
	Entity Linking	Identifying entity mentions in text and linking them to a knowledge base [39, 70, 78]
	Formal translation	Convert natural language text into formal representations (e.g., SQL, logical forms) [22, 23, 62, 89]
	Recommendation	Suggest relevant items (e.g., products, documents, movies) [45, 90]
	Action execution	Call APIs based on pre-generated requests [64, 84]
	Verification	Validate model outputs for consistency, logic, and constraint compliance [23, 52, 89]
	Aggregation	When multiple answers available, select/aggregate into one [13, 52, 95, 96]
	GIO creation	Create a refined, synthesized output that enhances final response for better user interaction; i.e., Generated Information Object (GIO) [16]
	<i>Complex</i>	
	RAG	Combine retrieval and generation to produce factually grounded responses [38]
	IRCoT	Integrate retrieval with chain-of-thought for improved information synthesis [69]
	Self-RAG	Generate text interleaved with reflection tokens to critique and guide the generation process in real-time [6]
	LINC	Integrate translation to formal representation and verification for complex logical queries [52]
	CoT-decoding	Elicit intrinsic CoT reasoning in LLMs by exploring top-k alternative tokens during decoding [72]
Executors	<i>Agents</i>	
	Foundation Models and LLMs	Specific LLMs such as GPT [10, 54, 55], Llama [68], T5 [14], and multi-modal models [4, 17, 58, 59] at different scales and reasoning capabilities, or trained with various techniques
	<i>Tools</i>	
	Retrieval models	Retrieving pieces of information (document text, KG triples, images, etc.) [34, 44, 61]
	Symbolic solvers	Tools that verify logical statements or prove the validity of expressions using formal reasoning [18, 49]
Resources	Other APIs	External APIs used for various tasks beyond retrieval and reasoning [84]
	Document collections	Internal/external collections of documents, reports, articles, and web pages, possibly in different modalities
	Semi-structured resources	Knowledge graphs with semi-structured sources of factual data (e.g., Wikidata) and databases with predefined schema (e.g., relational SQL databases) [7]

specific modality in a response, or task modules may require one or several modalities, e.g., *text, images, sound, video*, etc.

**Availability: Public, private, or proprietary.** Finally, the availability of a resource should also be considered. Ideally a resource is publicly available with no further costs, e.g., *Wikipedia* and *DBpedia* [7]. However, some resources are private and can only be used for a single user or group of users, e.g., the *user’s conversation log* or a *personal* or *corporate document collection*. Finally, proprietary resources could incur an additional monetary cost if accessed; a further consideration when deciding on a pipeline.

### 4.3 Dynamically creating system pipelines

The final part of our framework is a methodology that chooses and connects modules to construct a pipeline. We formulate a pipeline as a graph where nodes are modules and edges represent the interactions between them. Uniquely, as per our perspective in Section 3, we construct a new pipeline for every incoming query on-the-fly, such that our system design is dynamic and adaptive. We propose that this construction phase is best approached as a planning or multi-step decision process. Accordingly, we posit that planning and reinforcement learning methods appear to be the best solutions for optimizing the construction phase. As their design matches the task of learning the best sequence of tasks, assignment of executors, and allocation of resources to optimize an objective trade-off.

#### 4.3.1 Graph-based representation of pipelines

Taking inspiration from Zhuge et al. [96], we represent each pipeline as a directed acyclic graph (DAG). Let  $\mathcal{G}$  denote the set of all feasible pipeline graphs. Each candidate pipeline  $G \in \mathcal{G}$  is a subgraph of the overarching module graph:

$$G = (V, E), \quad \text{where } V \subseteq \mathcal{T} \cup \mathcal{E} \cup \mathcal{R}, \quad (1)$$

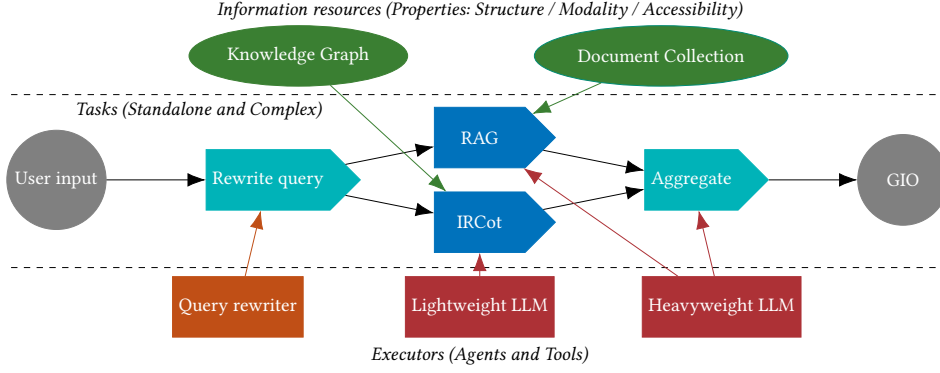
and the directed edges  $e \in E$  capture: (a) the sequence and flow of information among task modules, and (b) the assignment/allocation relationships between tasks, executors, and resources.

Graph configuration appears well-suited for our framing of pipeline construction as a planning problem. Since, in this framing, constructing a pipeline can amount to sequentially adding nodes and edges until a complete and valid graph is obtained.

#### 4.3.2 Query context features

Our aim is to re-construct and adapt the system’s pipeline for each incoming user query. This requires contextual features that are informative about what designs suit a query. Following contextual bandit terminology, we use a *context vector*  $\mathbf{x}_t$  to represent these features for the query of timestep  $t$ . It is crucial when defining  $\mathbf{x}_t$  to consider what aspects of the query the system should adapt to. We categorize potential choices for features into three groups:

- (1) *Representations of the user query*: Features regarding the user’s input query (text) encoding semantic and syntactic aspects



**Figure 1: Example of a graph that represents a system design for a possible pipeline in our framework. Nodes represent modules and are grouped according to their type: *Tasks*, *Executors* and *Resources*, in addition to nodes for the *user's input* and the *output's presentation* (GIO [16]). Edges between tasks represent their order of execution and dependencies, edges from executors indicate which are used to execute tasks, edges from resources indicate which are used for tasks.**

or more specific characteristics such as linguistic complexity, indicators of sentiment or emotion, and overall thematic or topical information. These should give an indication of query complexity which seems vital for estimating what effectiveness-efficiency trade-off can be made by a pipeline.

- (2) *User profile features*: This category includes data that characterize the user, such as individual preferences or relevant historical patterns or personal data [33]. These features can reduce ambiguity by contextualizing queries, e.g., the typical complexity of a user's questions, or resources they often need [71].
- (3) *Implicit behavior signals*: User behavior and interaction patterns often provide a lot of implicit information about the user's current session. For instance, preferences are rarely directly stated but can often be inferred from user interactions. Potentially, future systems may even use facial expressions or brain signals to enhance query understanding and provide further information about users' needs [28, 29, 50, 80, 81].

Together, these features should be indicative of what tasks and resources are needed to respond to the query, efficiently or effectively.

#### 4.3.3 The objective of pipeline construction

The pipeline construction aims to identify a configuration  $G \in \mathcal{G}$  that optimally balances efficiency and effectiveness. Formally, we define a composite objective function  $J(G) \in \mathbb{R}$  as a weighted combination of multiple reward and cost signals. For the optimization to work, an increase in the objective should indicate a better model, i.e.,  $J(G) > J(G')$  indicates  $G$  is preferred over  $G'$ . We identify four primary sources of feedback on which objectives can be build:

- (1) *User behavior and implicit feedback*: User interactions serve as implicit optimization signals. Additionally, explicit user annotations, i.e., direct indications of satisfaction with a system response, can give further guidance for improvements.
- (2) *Supervised examples*: Examples with ground-truth labels, such as manually annotated query resolution pairs or curated gold-standard responses, can be used as target outputs. A corresponding objective can measure similarity with the ground-truth as a measure of system effectiveness.
- (3) *Self-supervised objectives*: Techniques for self-supervision aim to optimize models for goals that do not require labels, but still correlate with effectiveness. In our example, a very powerful

but inefficient pipeline could provide an example of effectiveness that can be used as feedback to optimize a more efficient pipeline, similar to model distillation [35]. Furthermore, certain models (e.g., most foundation models) can provide confidence scores about their predicted response. Such signals can serve as guidance on the efficiency-effectiveness trade-off of a system [20]. For example, low confidence could be an indication that less efficient module additions are required for a query.

- (4) *Cost considerations*: Efficiency constitutes an important dimension of our framework's objective, and thus, requires a chosen measure of cost. Potential costs to consider include: (a) *Response time* or latency of the pipeline, i.e., how many milliseconds do users have to wait for a response to their query; (b) *Computational costs* of the system, measured in resources like memory, CPU/GPU/TPU FLOPS and network usage. (c) *Environmental impact* of the pipeline, i.e., its carbon footprint, fresh water usage, etc. [63]. (d) *Financial costs* stemming from API calls, hardware usage or proprietary resource access. These costs are interrelated, and while ideally they would be measured directly, we may have to depend on estimates only.

In addition to average costs, minimizing the variance of these cost measures can enhance robust pipeline behavior in practice. We note that feedback signals are rarely perfect; one should consider problems of (selection) bias stemming from feedback sources and transformations from feedback to objectives [25, 53].

#### 4.3.4 Methodologies for pipeline construction

A core feature of our proposed framework is the automatic pipeline construction for each individual user query. This requires a methodology for two core tasks: constructing pipeline graphs on-the-fly based on contextual features, and learning what pipelines work best for which queries. Thereby, the framework achieves adaptability through selecting and assembling tasks, executors, and resources tailored to individual queries. Given the combinatorial nature of possible pipelines, we propose that the best suited methodologies come from classical planning and reinforcement learning (RL):

- (1) *Classical planning algorithms*: In scenarios where the effectiveness and efficiency of (partial) pipelines can be estimated accurately, pipeline construction can be approached as a search over possible graph configurations. Adding a module or edge can

be seen as traversing a tree of possible graphs. Heuristic and graph-based search methods such as  $A^*$ , *beam search*, and *Monte Carlo tree search* could be effective to determine the optimal configuration [11].

- (2) *General reinforcement learning*: In order to apply RL approaches, the pipeline construction process has to be formulated as a Markov Decision Process (MDP). For instance, *states* represent partially constructed pipelines and the query context features; *actions* are the addition of a module nodes or edges; and *rewards* can come from the objective function (see Section 4.3.3). General RL methods can then be applied to the MDP, e.g., *REINFORCE* [75], *Q-learning* [73] or *actor-critic* [36] approaches.
- (3) *Contextual bandit algorithms*: In scenarios where the number of possible graphs is limited, contextual bandit algorithms become relevant. This subcategory of RL algorithms works for cases where the number of actions is discrete and rewards are immediate and only depend on single actions. Whilst these methods only apply in a subspace of all possible settings for RL, their specialization enables them to learn from far fewer examples than general RL methods, thus allowing for increased adaptivity. If the number of modules is limited, one can pre-compute all possible graphs and use them as the set of actions for CMAB algorithm.

#### 4.4 Limitations and potential extensions

Before diving into a specific instantiation of our framework, using CMAB, we consider future extensions that would address limitations of our proposal.

To start, reliance on RL methods that respond to contextual features comes with two possible limitations. First, the data-efficiency of RL methods could be too low to be responsive, i.e., if the RL method is too slow to learn a new graph construction strategy, it may already have been outdated before it is learned. This problem is most likely when the number of possible graphs is enormous, since RL methods have difficulty with large action spaces. Second, the contextual features may not be sufficiently informative to enable effective adaptivity, i.e., the query context features may not enable the identification of the best graph structure. Both are general problems with the RL methodology, where promising advancements in the field of RL may help alleviate these limitations [9, 42].

Also, our framework is designed to construct a graph for each query for a given set of modules. The optimization for underlying modules is taken for granted, and the constructed graphs are only based on features that are available when a queries are issued. Obvious directions of extension include the joint optimization of graph construction and underlying modules, e.g., by incorporating prompt and model fine-tuning in the learning process; and, enabling graphs that are dynamic during execution. The latter would enable graphs that adapt to intermediate outcomes, e.g., a first task could be to retrieve a result, and the subsequent tasks are chosen based on that first result. Whilst this would greatly increase the complexity of the graph construction process, it could be a logical continuation for development of our framework.

## 5 An Instantiation of our Framework with a Contextual Bandit Approach

To illustrate how our framework could be applied, we provide an example instantiation for a question-answering (QA) task.

### 5.1 Dataset

We use the dataset developed by Jeong et al. [27] for building and evaluating RAG for QA systems. It features a diverse set of questions, each labeled for one of three levels of complexity using Flan-T5-XL. Label A indicates that the question can be answered directly without retrieval; label B requires single-step retrieval; and label C necessitates multiple reasoning and retrieval steps. For our study, we randomly select 210 training and 51 test questions, ensuring a balanced distribution across complexity levels for optimization.

### 5.2 Graph definition

Our choice of nodes and possible edges matches the setup proposed by Trivedi et al. [69] and used by Jeong et al. [27]. We define three complex tasks and one standalone task:

$$\mathcal{T} = \{\text{NoR}, \text{OneR}, \text{IRCoT}, \text{Aggregate}\}. \quad (2)$$

- **NoR**: Answer generation with *no retrieval* augmentation, to be generated by an LLM agent module. The NoR task is to answer questions without additional external data, making it most suited for simple non-knowledge-intensive questions that match its encoded parametric knowledge.
- **OneR**: Answer generation with *one-time retrieval*, to be generated by an LLM agent module with a retriever tool and resource. During this task, a retriever tool is called *once* to provide seven potentially relevant documents to the executor. OneR is suited for questions that require access to external knowledge.
- **IRCoT**: Answer generation with following the IRCoT procedure, performed by an LLM agent module with a retriever tool and resource. Retrieved documents are interleaved in several back-and-forth chain-of-thought (CoT) reasoning steps with the LLM. As the most complex and time-consuming strategy, IRCoT is most-suited for questions where NoR and OneR are expected to fail, i.e., ones that demand extensive knowledge or a synthesis of multiple pieces of knowledge to produce an accurate answer.
- **Aggregate**: Aggregating the answers from multiple tasks into a single coherent answer, to be performed by an LLM or aggregator tool, i.e., a simple rule-based function.

The *NoR*, *OneR*, *IRCoT* tasks can only be executed in parallel, the *Aggregate* task can solely process the output of the other tasks.

For executor modules, we include Flan-T5-XL as an *LLM agent*, and BM25 as a *retrieval tool module* [60], and an *aggregator tool* that applies a majority voting function to select the most frequent answers from connected tasks. We avoid LLM-based aggregators as they may introduce errors or hallucinations.<sup>2</sup>

Finally, the resource module includes two corpora: *Wikipedia* [34] and a *passage corpus* constructed for multihop questions [69].

<sup>2</sup>See [95, 96] for other possible aggregation modules and their purposes.

**Table 3: Performance of each individual agent on the training set (by F1-score and average time in seconds).**

	NoR		OneR		IRCoT	
	F1	Time (s)	F1	Time (s)	F1	Time (s)
<b>Context A</b>	0.914	0.66	0.677	6.46	0.730	189.78
<b>Context B</b>	0.061	0.66	0.518	7.34	0.580	192.30
<b>Context C</b>	0.066	0.67	0.146	6.41	0.458	184.85
<b>Overall</b>	0.347	0.66	0.447	6.74	0.589	188.97

### 5.3 Optimization with CMAB

Due to the limited number of possible valid graphs in our setup (seven in total), we choose to apply the LinUCB contextual multi-armed bandit (CMAB) algorithm [41]. Our contextual features are the complexity labels provided in the dataset that are indicative of what kind of pipeline is required to answer each question. The reward function is a linear combination between *correctness* ( $P_t$ ), evaluated by F1-score, and costs from compute time ( $T_t$ ):

$$r_t = \beta \cdot P_t - (1 - \beta) \cdot T_t, \quad (3)$$

we set  $\beta = 0.5$  and  $T_t = S_t \left( \frac{\mathbb{1}[1 < S_t \leq 10]}{10000} + \frac{\mathbb{1}[S_t > 10]}{50} \right)$  where  $S_t$  is execution time in seconds. Our choice of  $T_t$  represents that we do not care for times below a single second, and really want to avoid times over ten seconds. We start with a pre-computing phase where every valid graph is generated and gathered in a set, this results in seven valid graphs that serve as the arms of the CMAB. Subsequently, the CMAB process is started: at each timestep  $t$  a random question is sampled from the training set, based on its context features CMAB chooses an arm which determines the system pipeline used to answer the query. The reward for the resulting answer is computed and CMAB updates its parameters accordingly, and its parameters are evaluated on the test-set. This process is repeated for 3,500 timesteps, during which CMAB aims to learn which system pipelines best match which question complexities.

### 5.4 Experimental results

Here we present results for the instantiated problem in Section 5. To interpret the results of our CMAB experiments effectively later, we first assess the individual performance of each Task defined in  $T$  on the training dataset. As shown in Table 3, performance varies by complexity of inputs: NoR excels in simple cases, while IRCoT dominates at higher complexities (B and C). At level B, IRCoT slightly outperforms OneR (.580 vs. .518) but incurs significantly higher latency (192.30 vs. 6.41s) due to iterative retrieval and reasoning. This trade-off highlights the practical cost of improved accuracy, as increased response time translates to the waiting time for the user to get a potentially correct response from the system.

#### 5.4.1 Can CMAB learn to map query complexity to optimal pipeline structures?

We aim to verify whether CMAB can learn to adaptively map different levels of input complexities to the optimal graph configurations.

- (a) **Time-agnostic reward.** In Figure 2, the top row depict the expected rewards for LinUCB, without considering time in the reward. These results reveal optimal actions identified for each complexity label; NoR for context A, and IRCoT for contexts

**Table 4: Evaluation of AQA (NT: Time-agnostic reward, T: Time-based reward) and GPTSwarm [96] on the test set by F1-score and time (log-transformed, in ms). Both have been trained on the training set. For GPTSwarm, the final optimized graph configuration is used for evaluation.**

	AQA (NT)		AQA (T)		GPTSwarm	
	F1	Time	F1	Time	F1	Time
<b>Context A</b>	1.0	6.18	1.0	6.18	0.862	12.78
<b>Context B</b>	0.568	12.04	0.539	8.73	0.327	12.79
<b>Context C</b>	0.523	11.75	0.523	11.75	0.317	12.76
<b>Overall</b>	0.697	9.99	0.687	8.89	0.502	12.78

B and C, based on F1-scores. The dotted lines depict the real reward calculated for the best possible action per context.

- (b) **Time-based reward.** The bottom row of Figure 2 depicts results for training while including time cost in the reward calculation. Considering both performance and the time cost, the real reward for best actions per label are shown in the plots as dotted red lines. Including time in the reward calculation, we observe the model preferentially selects pipelines that balance efficacy with reasonable execution times. In context B, it prioritizes the configuration with only OneR over configurations with IRCoT or a combination of better-performing agents (i.e., OneR+IRCoT).

We conclude that our approach successfully constructs system pipelines that are adapted to the characteristics of incoming questions and optimized for a given efficiency-effectiveness trade-off.

#### 5.4.2 Can adaptive orchestration achieve a superior trade-off over static optimization?

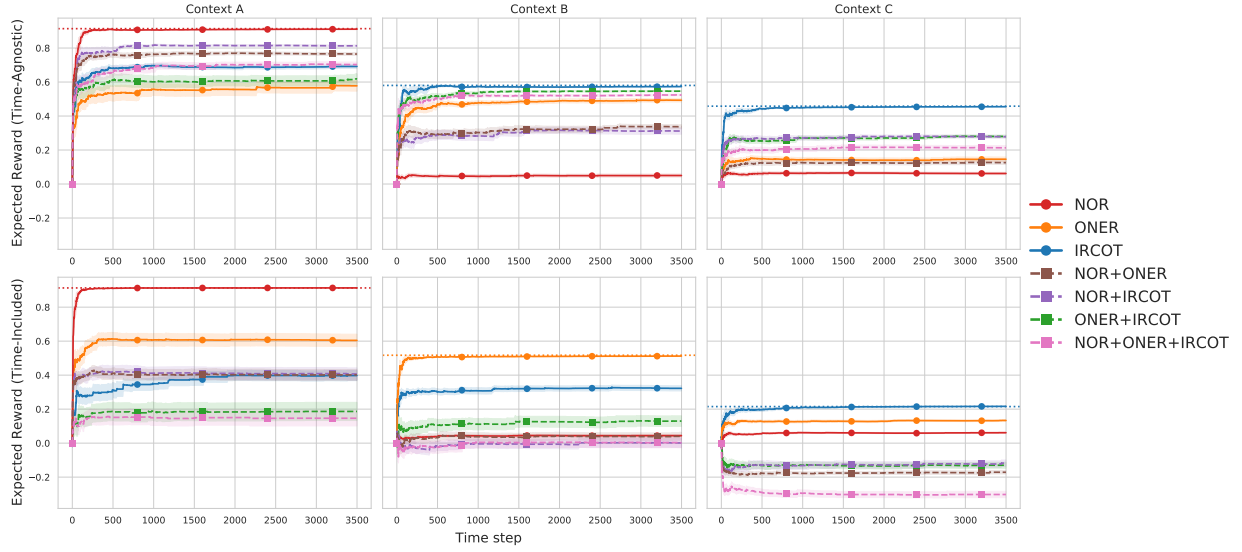
To demonstrate the importance of adaptivity in designing an orchestration framework, here, we aim to assess how a non-adaptive orchestration optimization affects the performance. To this aim, we train our *adaptive question-answering model* (AQA) and GPTSwarm [96] and assess them on our test set. Both models are given the set of modules described in Section 5.2. Since GPTSwarm does not accommodate context or time cost in their proposed framework, in our experiment with this framework we feed each question to the model and the optimization gradually converges to the *most optimal* graph configuration by optimizing towards better performance (F1-score).<sup>3</sup>

Prior to optimization with REINFORCE, the graph maintains a uniform edge probability distribution. After 200 training epochs, optimization converges on a structure where only OneR and IRCoT nodes retain edges, while OneR's connection to the final decision node is pruned out (following the thresholding strategy of [96], edges with probabilities below .5 are removed). This shows, as expected, that this static framework fails to adapt to different complexity levels by routing for a good enough answering strategy that is also time-efficient, as it only favors higher accumulative F1 scores across time.

The evaluation results on the test set are shown in Table 4. The first two columns show the CMAB model (LinUCB) with and without time considered in the reward implementation respectively and

<sup>3</sup>Similar to what is done in their MMLU experiment [96].





**Figure 2: LinUCB expected rewards for the collaborative action space, dashed line depicts real reward for the optimal action.**

the last column is the final optimized graph using REINFORCE. Using a fixed orchestration by GPTSwarm, we observe a fall in performance for different complexity levels, as it is not possible to adapt the strategy based on the characteristics of the question, which leads to a lower overall performance compared to AQA (NT and T). Also, as the fixed configuration in GPTSwarm is also more sophisticated (i.e., both NoR and IRCOT have edges to the final decision node), the time cost is constantly higher compared to AQA.

## 6 Summary and Future Research Directions

In this paper, we recognize that the landscape of generative information access systems is marked by the proliferation of heterogeneous modules and tools – each designed to serve similar purposes, yet emerging as isolated implementations. Observing this fragmented state, we raised and addressed critical questions regarding the design of a comprehensive framework that allows for the inclusion of these heterogeneous modules. We posit that in order to move forward, module orchestration is not enough; the field needs to develop methods for *adaptive* module orchestration. The ultimate goal should be that the optimal answering strategy can be selected for each user input, so that the most relevant answer is inferred at the lowest possible cost. In this regard, we proposed a future-ready framework that defines the potential modules (i.e. building blocks) of a GenIA system as nodes of a graph with dynamic edges, enabling adaptive orchestration of pipelines in real-time.

Our perspective bridges existing gaps by introducing a formalized approach to adaptive orchestration, ensuring that chosen answering strategies are context-aware and resource-efficient. To realize such a comprehensive self-organizing adaptive system built on many subcomponents, we have outlined how such a framework and its optimization can be modeled. To indicate the practical usage of our framework, we provided a proof-of-concept instantiation of this framework for a query-answering task. Our experimental evaluation of this instantiation indicated that our approach can successfully construct system pipelines that are adapted to the characteristics of incoming questions and optimized for a given efficiency-effectiveness trade-off.

Looking ahead, we like to pose some open questions for the community and some future research directions. (i) As the system scales by integrating more modules, how should optimization methods adapt to ensure efficiency and robustness? (ii) What internal signals from system subcomponents (e.g., uncertainty or likelihood signals) can enhance decision-making at each step? How should these signals be leveraged effectively? (iii) How to handle orchestration complexity that arises from more granular node definitions and task decompositions? (iv) How can optimization models be designed to directly learn dependencies between context features and graph configurations, ensuring adaptive orchestration without relying on intermediate classification/clustering of the queries? (v) How and what additional contextual signals should inform decision-making? (vi) How can we account for other levels of optimization (e.g., enhancing prompts’ quality or fine-tuning at individual module level)? (vii) As the number of optimization goals increases, how can we ensure effective convergence without compromising performance? These questions opens up interesting research directions and new challenges, that may bring us closer to achieving self-organizing conversational assistants that automatically adapt to different environments while maintaining robustness and flexibility.

## Resources

Resources needed to reproduce the results of this paper are publicly available at <https://github.com/informagi/AQA>.

## Acknowledgments

We thank Nik Vaessen for his intellectual contributions to help shape our ideas. This research is supported by the Dutch Research Council (NWO) under project numbers NWA.1389.20.183, VI.Veni.-222.269, 024.004.022, and KICH3.LTP.20.006, and the EU’s Horizon Europe program under grant No. 101070212 and No. 101070014 (OpenWebSearch.EU, <https://doi.org/10.3030/101070014>). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

## References

- [1] Leonard Adolphs, Benjamin Börschinger, Christian Buck, Michelle Chen Huebscher, Massimiliano Ciaramita, Lasse Espeholt, Thomas Hofmann, Yannic Kilcher, Sascha Rothe, Pier Giuseppe Sessa, and Lierni Sestorain. 2022. Boosting Search Engines with Interactive Agents. *Trans. Mach. Learn. Res.* (2022).
- [2] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2021. Building and Evaluating Open-Domain Dialogue Corpora with Clarifying Questions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.
- [3] James Allan, Eunsol Choi, Daniel P Lopresti, and Hamed Zamani. 2024. Future of Information Retrieval Research in the Age of Generative AI. *arXiv preprint arXiv:2412.02043* (2024).
- [4] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A Family of Highly Capable Multimodal Models. *CoRR* (2023).
- [5] Gaurav Arora, Shreya Jain, and Srujana Merugu. 2024. Intent Detection in the Age of LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*.
- [6] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations, ICLR*.
- [7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference*.
- [8] Kristian G. Barman, Sascha Caron, Emily Sullivan, Henk W. de Regt, Roberto Ruiz de Austri, Mieke Boon, Michael Färber, Stefan Fröse, Faegheh Hasibi, Andreas Ipp, Rukshak Kapoor, Gregor Kasieczka, Daniel Kostić, Michael Krämer, Tobias Golling, Luis G. Lopez, Jesus Marco, Sydney Otten, Pawel Pawlowski, Pietro Vischia, Erik Weber, and Christoph Weniger. 2025. Large Physics Models: Towards a collaborative approach with Large Language Models and Foundation Models. *arXiv:2501.05382*
- [9] Jakob Bauer, Kate Baumli, Feryal M. P. Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collier, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Satinder Singh, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei M. Zhang. [n. d.]. Human-Timescale Adaptation in an Open-Ended Task Space. In *International Conference on Machine Learning, ICML 2023*.
- [10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. [n. d.]. Language Models are Few-shot Learners. In *Advances in neural information processing systems*.
- [11] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intell. AI Games* (2012).
- [12] H. Chase. 2022. LangChain. <https://github.com/hwchase17/langchain>. Accessed: 2025-02-07.
- [13] Zheng Chu, Jingchang Chen, Qianglong Chen, Haotian Wang, Kun Zhu, Xiyuan Du, Weijiang Yu, Ming Liu, and Bing Qin. 2024. BeamAggR: Beam Aggregation Reasoning over Multi-source Knowledge for Multi-hop Question Answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- [14] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2024. Scaling Instruction-Finetuned Language Models. *J. Mach. Learn. Res.* (2024).
- [15] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonello, and Fabrizio Silvestri. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [16] J. Shane Culpepper, Fernando Diaz, and Mark D. Smucker. 2018. Research Frontiers in Information Retrieval: Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018). *SIGIR Forum* (2018).
- [17] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. 2024. Vision Transformers Need Registers. In *The Twelfth International Conference on Learning Representations, ICLR 2024*.
- [18] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS*.
- [19] Yang Deng, Lizi Liao, Zhonghua Zheng, Grace Hui Yang, and Tat-Seng Chua. 2024. Towards Human-centered Proactive Conversational Agents. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*.
- [20] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nat.* (2024).
- [21] Shahla Farzana, Qunzhi Zhou, and Petar Ristoski. 2023. Knowledge Graph-Enhanced Neural Query Rewriting. In *Companion Proceedings of the ACM Web Conference 2023*.
- [22] Wenlong Fei, Xiaohua Wang, Min Hu, Qingyu Zhang, and Hongbo Li. 2024. MTLS: Making Texts into Linguistic Symbols. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.
- [23] Debargha Ganguly, Srinivasan Iyengar, Vipin Chaudhary, and Shivkumar Kalyanaraman. 2024. PROOF OF THOUGHT : Neurosymbolic Program Synthesis allows Robust and Interpretable Reasoning. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.
- [24] Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. 2024. Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks. *CoRR abs/2407.21059* (2024).
- [25] Shashank Gupta, Philipp Hager, Jin Huang, Ali Vardasbi, and Harrie Oosterhuis. 2023. Recent Advances in the Foundations and Applications of Unbiased Learning to Rank. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [26] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiwu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations, ICLR*.
- [27] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL*.
- [28] Kaixin Ji, Danula Hettiachchi, Flora D. Salim, Falk Scholer, and Damiano Spina. 2024. Characterizing Information Seeking Processes with Multiple Physiological Signals. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2024)*.
- [29] Kaixin Ji, Damiano Spina, Danula Hettiachchi, Flora Dilys Salim, and Falk Scholer. 2023. Examining the Impact of Uncontrolled Variables on Physiological Signals in User Studies for Information Processing Activities. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [30] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. *CoRR* (2024).
- [31] Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). 7969–7992.
- [32] Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. *CoRR* (2024). To appear in the Web Conference 2025.
- [33] Hideaki Joko, Shubham Chatterjee, Andrew Ramsay, Arjen P De Vries, Jeff Dalton, and Faegheh Hasibi. 2024. Doing personal laps: Llm-augmented dialogue construction for personalized multi-session conversational search. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 796–806.
- [34] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*.
- [35] Pooya Khandel, Andrew Yates, Ana Lucia Varbanescu, Maarten de Rijke, and Andy D. Pimentel. 2024. Distillation vs. Sampling for Efficient Training of Learning to Rank Models. In *Proceedings of the 2024 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR*.

- [36] Vijay R. Konda and John N. Tsitsiklis. 1999. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems 12, [NIPS Conference]*.
- [37] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *9th International Conference on Learning Representations, ICLR*.
- [38] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*.
- [39] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient One-Pass End-to-End Entity Linking for Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- [40] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS*.
- [41] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 661–670.
- [42] Wenzhe Li, Hao Luo, Zichuan Lin, Chongjie Zhang, Zongqing Lu, and Deheng Ye. 2023. A Survey on Transformers in Reinforcement Learning. *Trans. Mach. Learn. Res.* 2023 (2023).
- [43] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. In *The Twelfth International Conference on Learning Representations, ICLR*.
- [44] Zihao Li, Yuyi Ao, and Jingrui He. 2024. SpherE: Expressive and Interpretable Knowledge Graph Embedding for Set Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*. 2629–2634.
- [45] Defu Lian, Xu Huang, Xiaolong Chen, Jin Chen, Xingmei Wang, Yankai Wang, Haoran Jin, Rui Fan, Zheng Liu, Le Wu, and Enhong Chen. 2023. RecStudio: Towards a Highly-Modularized Recommender System. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [46] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS*.
- [47] Chaitanya Malaviya, Peter Shaw, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2023. QUEST: A Retrieval Dataset of Entity-Seeking Queries with Implicit Set Operations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*.
- [48] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khoshabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.).
- [49] W. McCune. [n. d.]. Prover9 and Mace4. [n. d.].
- [50] Niall McGuire and Yashar Moshfeghi. 2024. DEEPER: Dense Electroencephalography Passage Retrieval. *CoRR abs/2412.06695* (2024).
- [51] Marvin Minsky. 1988. *Society of mind*.
- [52] Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.
- [53] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-Aware Unbiased Learning to Rank for Top-k Rankings. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR*.
- [54] OpenAI. 2023. OpenAI O1 System Card. <https://openai.com/index/openai-o1-system-card/>.
- [55] OpenAI. 2023. OpenAI O3 Mini System Card. <https://openai.com/index/o3-mini-system-card/>.
- [56] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Trans. Knowl. Data Eng.* (2024).
- [57] Andrew Parry, Debasis Ganguly, and Manish Chandra. 2024. "In-Context Learning" or: How I learned to stop worrying and love "Applied Information Retrieval". In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [58] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event*.
- [59] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event*.
- [60] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gattford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text Retrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2–4, 1994*.
- [61] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* (2009).
- [62] Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2025. Divide and Translate: Compositional First-Order Logic Translation and Verification for Complex Logical Reasoning. In *The Thirteenth International Conference on Learning Representations*.
- [63] Harrison Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, Reuse, Recycle: Green Information Retrieval Research. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [64] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS*.
- [65] Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic Retrieval Augmented Generation based on the Real-time Information Needs of Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 12991–13013.
- [66] Peter T. Szymanski and Michael D. Lemmon. 1993. Adaptive mixtures of local experts are source coding solutions. In *Proceedings of International Conference on Neural Networks (ICNN'88)*.
- [67] Xiaqiang Tang, Qiang Gao, Jian Li, Nan Du, Qi Li, and Sihong Xie. 2025. MBARAG: A Bandit Approach for Adaptive Retrieval-Augmented Generation through Question Complexity. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING*.
- [68] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* (2023).
- [69] Harsh Trivedi, Niranjana Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*.
- [70] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. 2020. REL: An Entity Linker Standing on the Shoulders of Giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. 2197–2200.
- [71] Hemanth Vemuri, Sheshansh Agrawal, Shivam Mittal, Deepak Saini, Akshay Soni, Abhinav V. Sambasivan, Wenhao Lu, Yajun Wang, Mehul Parsana, Purushottam Kar, and Manik Varma. 2023. Personalized Retrieval over Millions of Items. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [72] Xuezhi Wang and Denny Zhou. 2024. Chain-of-Thought Reasoning Without Prompting. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*.
- [73] Christopher J. C. H. Watkins and Peter Dayan. 1992. Technical Note Q-Learning. *Mach. Learn.* (1992).
- [74] Orion Weller, Dawn J. Lawrie, and Benjamin Van Durme. 2024. NevIR: Negation in Neural Information Retrieval. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL*.
- [75] Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* (1992).
- [76] Ian Wu, Sravan Jayanthi, Vijay Viswanathan, Simon Rosenberg, Sina Pakazad, Tongshuang Wu, and Graham Neubig. 2024. Synthetic Multimodal Question Generation. In *Findings of the Association for Computational Linguistics: EMNLP*.
- [77] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *CoRR abs/2308.08155* (2023).
- [78] Zilin Xiao, MING GONG, Jie Wu, Xingyao Zhang, Linjun Shou, and Daxin Jiang. 2023. Instructed Language Models with Retrievers Are Powerful Entity Linkers. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- [79] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.).



- [80] Ziyi Ye, Qingyao Ai, Yiqun Liu, Maarten de Rijke, Min Zhang, Christina Lioma, and Tuukka Ruotsalo. 2025. Generative language reconstruction from brain recordings. *Communications Biology* 8, 1 (March 2025), 346.
- [81] Ziyi Ye, Jingtao Zhan, Qingyao Ai, Yiqun Liu, Maarten de Rijke, Christina Lioma, and Tuukka Ruotsalo. 2024. Query Augmentation with Brain Signals. In *Proceedings of the 32nd ACM International Conference on Multimedia*. Association for Computing Machinery.
- [82] Wenpeng Yin, Muhao Chen, Rui Zhang, Ben Zhou, Fei Wang, and Dan Roth. 2024. Enhancing LLM Capabilities Beyond Scaling Up. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*.
- [83] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. In *The Twelfth International Conference on Learning Representations, ICLR*.
- [84] Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. 2024. CRAFT: Customizing LLMs by Creating and Retrieving from Specialized Toolsets. In *The Twelfth International Conference on Learning Representations*.
- [85] Yifei Yuan, Clemencia Siro, Mohammad Aliannejadi, Maarten de Rijke, and Wai Lam. 2024. Asking Multimodal Clarifying Questions in Mixed-Initiative Conversational Search. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*.
- [86] Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. 2024. The Shift from Models to Compound AI Systems. <https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/>.
- [87] ChengXiang Zhai. 2024. Large Language Models and Future of Information Retrieval: Opportunities and Challenges. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [88] An Zhang, Yang Deng, Yankai Lin, Xu Chen, Ji-Rong Wen, and Tat-Seng Chua. 2024. Large Language Model Powered Agents for Information Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. Association for Computing Machinery.
- [89] Tianhua Zhang, Jiaxin Ge, Hongyin Luo, Yung-Sung Chuang, Mingye Gao, Yuan Gong, Yoon Kim, Xixin Wu, Helen Meng, and Jim Glass. 2024. Natural Language Embedded Programs for Hybrid Language Symbolic Reasoning. In *Findings of the Association for Computational Linguistics: NAACL*.
- [90] Jujia Zhao, Wenjie Wang, Yiyang Xu, Teng Sun, Fuli Feng, and Tat-Seng Chua. 2024. Denoising Diffusion Recommender Model. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*.
- [91] Wenxuan Zhou, Qiang Ning, Heba Elfardy, Kevin Small, and Muhao Chen. 2022. Answer Consolidation: Formulation and Benchmarking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- [92] Yingxue Zhou, Jie Hao, Mukund Rungta, Yang Liu, Eunah Cho, Xing Fan, Yanbin Lu, Vishal Vasudevan, Kellen Gillespie, and Zeynab Raeesy. 2023. Unified Contextual Query Rewriting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*.
- [93] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Quoc V. Le, James Laudon, et al. 2022. Mixture-of-experts with Expert Choice Routing. *Advances in Neural Information Processing Systems* (2022).
- [94] Yunhua Zhou, Guofeng Quan, and Xipeng Qiu. 2023. A Probabilistic Framework for Discovering New Intents. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- [95] Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R. Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, Louis Kirsch, Bing Li, Guohao Li, Shuming Liu, Jinjie Mai, Piotr Piekos, Aditya Ramesh, Imanol Schlag, Weimin Shi, Aleksandar Stanić, Wenyi Wang, Yuhui Wang, Mengmeng Xu, Deng-Ping Fan, Bernard Ghanem, and Jürgen Schmidhuber. 2023. Mindstorms in Natural Language-Based Societies of Mind. *CoRR* abs/2305.17066 (2023).
- [96] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. GPTSwarm: Language Agents as Optimizable Graphs. In *Forty-first International Conference on Machine Learning, ICML*.