CIExMAS: Closed Information Extraction using Multi Agent Systems

Max Lautenbach[1,2], Sven Hertling[1]

[1]University of Mannheim, Mannheim, Baden-Württemberg, Germany
[2]SAP SE, Walldorf, Baden-Württemberg, Germany

Abstract

Closed Information Extraction (cIE) has primarily depended on fine-tuned Transformer models, which require large training datasets. Despite large language models (LLMs) having demonstrated a reduced need for fine-tuning in other domains, their potential use and additionally the use of LLM-based multi-agent systems in cIE has not yet been investigated. In this paper, we present **CIE**x**MAS** (**C**losed **I**nformation **E**xtraction using **M**ulti **A**gent **S**ystems), a collection of multi-agent systems that can extract triples from text without prior fine-tuning, meeting the requirements of cIE. This paper provides an exploratory framework for testing a range of architectural setups and evaluating the general feasibility of multi-agent systems in this domain. The best architecture comprises specialised agents for triple extraction, uniform resource identifier mapping and hallucination prevention, supported by tools for integration with knowledge graphs. In our evaluation, CIExMAS achieved up to 80% of the performance of the state-of-the-art synthIE model, demonstrating that high-quality extraction can be achieved using LLM-based agents without prior fine-tuning. Furthermore, our results identify property extraction as the primary remaining challenge in multi-agent systems for cIE. As an initial study in this direction, CIExMAS lays the foundation for further research combining LLM-based agents and cIE.

Keywords

Knowledge Graph Generation, multi-agent systems, Closed Information Extraction

1. Introduction

With the rapid development of large language models (LLMs), the need for semantically rich data sources to equip LLMs with relevant, up-to-date knowledge is crucial to their success. However, a vast majority of enterprise knowledge remains siloed in unstructured documents, including enterprise documents, reports, or communication logs [1]. This opens up the research area of automating the process of populating knowledge graphs, which can provide the semantics of such data in a machine-readable format. To be interoperable with existing graphs or ontologies, the extracted triples must be mappable to a defined knowledge graph, as described in the field of closed information extraction (cIE).

Prior work in the field of cIE has already shown that fine-tuned language models (LMs) can generate such triples with an F_1-score of over 90% on Wiki-cIE [2]. In parallel, recent advancements in the field of open information extraction (oIE) could demonstrate the capability of LLMs to extract free-form triples out of text using fine-tuning [3], prompt-optimisation [4] or a single agent system [5]. While oIE leverages the generative flexibility of LLMs, cIE demands strict adherence to a pre-defined knowledge base. The usage of multi-agent systems, which have demonstrated superior performance within benchmarks and reviews in general domains [6, 7, 8], remains an open topic in the research on cIE. In this paper, we investigate the extraction accuracy of multiple non-fine-tuned LLM-based multi-agent systems on the task of cIE.

To address this question, we introduce **CIE**x**MAS** (**C**losed **I**nformation **E**xtraction using **M**ulti **A**gent **S**ystems), a group of cIE multi-agent systems. In our empirical approach, we adapted an iterative refinement approach [9] with human feedback and optimisation to improve our agentic system step by step. Therefore, we identified edge cases, adapted the prompting and tooling, and subsequently evaluated

ESWC'26: 5th Workshop on LLM-Integrated Knowledge Graph Generation from Text (TEXT2KG), May 10-14, 2026 | Dubrovnik, Croatia

max.lautenbach@sap.com (M. Lautenbach); sven.hertling@uni-mannheim.de (S. Hertling)
https://github.com/maxlautenbach (M. Lautenbach)

our advances on the test split using standard metrics, specifically the F_1-score on the generated triples. To evaluate the extent to which multi-agent systems can solve the task of cIE, we compared our results with two state-of-the-art models.

Our work contributes multiple architectures to solve cIE with multi-agent systems; approaches that can solve cIE with little adaptation work across other knowledge bases and domains; and a modular agentic architecture that can be easily expanded to be optimised or adapted to other knowledge graphs or ontologies. Our code, as well as other experiments, can be found on GitHub[1].

In Section 2, we outline the related work within the domain of cIE alongside the work done using LLMs in information extraction overall. In Section 3, we explain our CIExMAS development approach and the three architectures used in this paper. The evaluation setup, evaluation results, and their discussion are presented in Section 4. Finally, we conclude our findings and outline steps for future work in Section 5.

2. Related Work

The field of cIE, as conceptualised by Josifoski et al. [10], builds upon similar approaches in the field of oIE. In both fields, most approaches can be divided into pipeline and end-to-end approaches. Pipeline approaches typically extract entities and relations, then disambiguate them to map the results to a knowledge base. Such approaches are, for instance, CycleGT [11] or DISCIE [12]. Another research stream focuses on processing the task of oIE in a single step, using an end-to-end approach. End-to-end approaches have historically outperformed pipeline approaches because they mitigate the error propagation across pipeline components [13, 14]. Examples for this approach are REBEL [15], Seq2RDF [16] or N-gram attention [13].

CIE approaches mainly follow the idea of oIE approaches, but oIE approaches are mostly not directly applicable towards cIE, due to the strict adherence to a given knowledge base in cIE [10]. Therefore, the GenIE approach of Josifoski et al. [10] follows the REBEL pattern [15] by leveraging a fine-tuned Transformer model to generate triples. In contrast to REBEL, Josifoski et al. [10] restricted the tokens generated by the Transformer to identifiable surface forms of entities and properties in the underlying knowledge base. This paradigm was further advanced in synthIE [2] by utilising a synthetic dataset to fine-tune the Transformer.

Contrary to the conclusions of Trisedya et al. [13], DISCIE by Möller and Usbeck [12] demonstrated that a pipeline approach can achieve superior performance compared to GenIE. It is based on mention recognition, generating entity candidates, ranking them, and subsequently providing relation extraction. The approach employs a trained model for each step. With these more specific models, DISCIE demonstrated performance improvements over other end-to-end generative approaches in the field of cIE [12].

To our knowledge, approaches utilising LLMs or even agentic artificial intelligence (AAI) have been employed only in the oIE domain. In AutoRE, Xue et al. [3] showed that fine-tuned LLMs can outperform larger non-finetuned LLMs for relation extraction in documents. In contrast, Chen et al. [4] employed GAP. An approach for prompt engineering in the relation extraction domain that outperforms other approaches. Both approaches cannot be compared, as they were not evaluated on the same dataset.

In the area of AAI-based relation extraction, the only approach we are aware of is AgentRE [5]. This approach leverages memory and retrieval tools attached to a single agent to generate surface-form triples. With those tools, AgentRE had direct access to samples or relevant information in the knowledge graph, as well as a memory to improve processing. Nevertheless, AgentRE lacks a direct mapping to the knowledge graph, which restricts its use for cIE. It is also non-comparable to other oIE approaches, as it was tested on different datasets. Ultimately, AgentRE highlights a distinctive research gap in multi-agent systems for cIE, while supporting the hypothesis of the effectiveness of pipeline approaches in oIE and cIE, such as DISCIE.

[1] https://github.com/maxlautenbach/CIExMAS

The most common dataset in the area of relation extraction is REBEL [15]. This silver-grade dataset is generated by leveraging hyperlinks in Wikipedia articles to identify all possible relations among the entities they mention. Afterwards, a Transformer model predicts if the Wikipedia text entails the hypothesis of the detected triple.

Instead of generating triples out of the text, Josifoski et al. [2] generated the text out of the triples. Therefore, they sampled triples from Wikidata and generated Wikipedia-like sentences, building the Wiki-cIE dataset. This dataset will also be used in this work, as Josifoski et al. [2] already showed that models trained and evaluated on Wiki-cIE can achieve drastically higher F_1-scores, which can be traced back to the lower quality of REBEL [2].

3. CIExMAS Approach

We present the CIExMAS approach in four stages: (1) the conceptual framework, (2) the agent design, (3) the integrated tools, and (4) the evaluated multi-agent architectures.

3.1. Conceptual Framework of CIExMAS

Our solution, CIExMAS, follows the modular pipeline approach in the information extraction domain, but it adds agentic decision-making. In this framework, agents have the autonomy to extract entities and relations, or to perform disambiguation of such entities and relations, based on the context held within the shared agentic state.

CIExMAS builds upon previous work that already introduced Transformers into the cIE task. It also incorporates recent advancements in oIE and generative artificial intelligence (AI) research by evaluating (multi)-agent systems. By leveraging the emergent natural language understanding of LLMs within a modular agentic framework, we aim to solve cIE tasks without the necessity for domain-specific fine-tuning. Architecturally, this makes CIExMAS ontology-adaptable by design, as the extraction model is decoupled from the underlying ontology, unlike previous fine-tuned approaches.

3.2. General Agent Design

While CIExMAS evaluates various agent designs with different tool sets, all share a similar design approach. All agents operate on a shared state schema that is iteratively updated to maintain context and track extraction progress. In addition, various agents can have limited read and write access to their state contingent on their role. Further details are outlined in the agent architecture section.

As this paper focuses on agent tools, agent architectures and their effectiveness, prompt engineering was conducted manually. Nevertheless, the prompt followed Chain-of-Thought prompting in combination with in-context learning by example, as shown to be effective in earlier work [17, 18].

In addition, the prompts were designed iteratively using a development split of Wiki-cIE. Therefore, we followed Hadfield et al. [19] in detecting errors early with small examples. In addition, the prompts were designed to be well-structured to decompose into human-executable subtasks, as proposed by Anthropic [20].

The agents are designed to use specific XML tags to manipulate their state or hand off to another agent. Furthermore, agents must produce correct Turtle output when they stop their iteration. While ensuring valid Turtle syntax can be non-trivial for LLMs, this strict requirement is enforced to guarantee downstream interoperability. In addition, the tools themselves will throw an error if an agent's input is faulty or if there is a processing error.

To enhance system reliability, we implemented a dedicated robustness layer with error propagation, enabling agents to self-repair their execution. This error handling includes a simple regex check of the required inputs and will loop back to the agent sending the message with a semantic-rich error, such as "No Turtle string provided in tool input or messages". In addition, errors coming from tools will be handed over to the agents as they are.

3.3. Agent Tools

As outlined, our CIExMAS approaches follow a modular pipeline for cIE, including a mapping to the knowledge base. Therefore, we use a tool called *uniform resource identifier (URI) Retrieval Tool*. This tool provides the agent with a similarity search over the vector-indexed labels of URI entries in Wikidata.

In general, the tool accepts free-form search terms from the agent. This has the advantage of allowing the agent to vary with search terms. In addition, the agents could include more context for searching. Nevertheless, through the concept of embeddings [21], most of the variation between the search terms could be assumed to be covered.

To leverage the flexibility of AI agents, we introduce three search methods. The first two are entity and property filtering. The third one is called example search. It will use embeddings for sentence-similarity search, using examples triples of a property from Wikidata. Therefore, the agent could either take the extracted free-form triples for a similarity search or use another arbitrary sentence. This facilitates context-aware property retrieval by bridging the semantic gap between textual mentions and formal property definitions.

The output of the *URI Retrieval Tool* itself contains the top three search results by cosine similarity alongside the description of an entity or property. Whenever a property is retrieved, an example triple including the respective property will also be included. The *URI Retrieval Tool* is designed as a simple similarity search tool. Therefore, it will not check any semantics provided by the underlying knowledge base.

To ensure the quality of the URI mapping, we introduce a *Semantic Validation Tool*. The *Semantic Validation Tool* will take Turtle as input and check the domain and range constraints provided by the underlying knowledge base. This is done by using SPARQL queries to check whether the type or the super-type of a subject or an object matches the property's restrictions. This result is looped back to the agent as structured text, including the check result and the expected property types.

Since our various agentic systems are proposed to produce Turtle as their final output, it is challenging for the LLMs to determine whether they have hallucinated. This is mainly due to the URIs used in the knowledge base. For instance, Wikidata uses a letter followed by numbers, such as Q141488. To provide the agent with a hallucination check, we propose a *Turtle-to-Label* tool. This tool returns a label resolution for each component of the input Turtle.

Combining the *URI Retrieval Tool* with the result-checking tools of *Semantic Validation* and *Turtle-to-Label*, we propose an agentic toolset to proceed with the cIE task. This includes enhancing entity and property disambiguation, followed by checking the produced outputs. With the idea of AAI, this leads to self-improving agent design.

3.4. Agent Architecture

In CIExMAS, we follow three agentic patterns: A supervisor pattern incorporating a main and multiple subagents; a ReAct agent based on the toolset presented; and a network pattern with three domain expert agents. The architectures represent an increasing level of complexity and agentic autonomy, starting from a hierarchical baseline to a collaborative expert network.

The supervisor architecture shown in Figure 1 was the first one implemented. The arrows in the figures show the allowed process flows in the agentic architecture. In the supervisor architecture, the allowed process flow follows the idea of enabling a straight cIE pipeline with the capability to loop over subagents' results. So the supervisor architecture consisted of one *Supervisor Agent*, an *Entity Extraction Agent* and a *Triple Extraction Agent*. Tool-wise, the supervisor architecture used the *URI Retrieval Tool*, which was directly connected to the *Supervisor Agent*. This has been kept since the beginning to design a baseline architecture that includes all required components to fulfil cIE, but no more. Therefore, the validation tools are not used within the supervisor architecture.

This design followed the idea of creating a domain expert for extracting entities and a domain expert for bringing those entities into triples, along with their properties. In the supervisor architecture, we use a simple enhancement to the *URI Retrieval Tool*, as the search terms defined by the *Supervisor Agent*

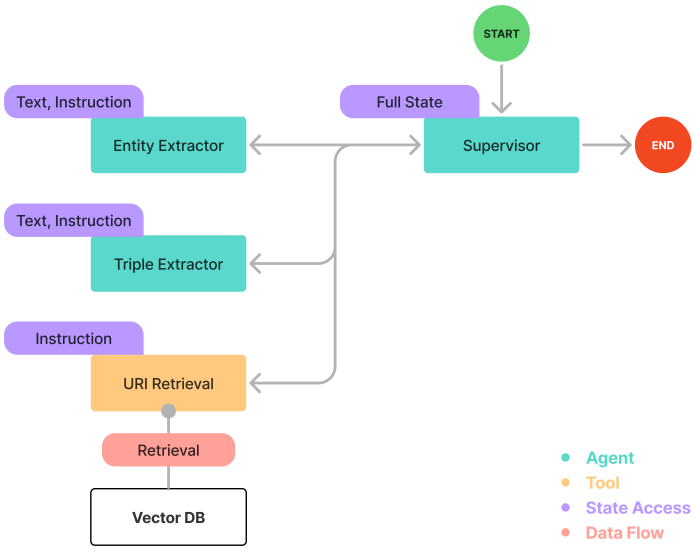

Figure 1: CIExMAS Supervisor Architecture

and processed by the *URI Retrieval Tool* were post-processed by another LLM call. This post-processing included a LLM-based semantic check and the formatting of the results from the *URI Retrieval Tool*.

The supervisor architecture is based on a simple agentic state. It consists of the `input text`, the `message history` and the `instruction` for either the tool or the agent as a string. The *Supervisor Agent* is the only one with access to the full state, while the *Entity Extraction Agent* and *Triple Extraction Agent* had access to both text and instructions.

The ReAct architecture shown in Figure 2, in contrast, is based on a single powerful agent, as proposed by Yao et al. [22]. In addition to the supervisor architecture, it also has access to the *Semantic Validation* and *Turtle-to-Label* tools. The design of a single powerful agent also risks role confusion of the agent, as outlined by OpenAI [23]. Therefore, the respective validation tools are used to ensure the quality of the ReAct architecture.

The idea behind the ReAct architecture is to test whether the simplest agentic solution can achieve comparable performance to other architectures while keeping resource requirements low. Therefore, the state is also very sleek, keeping the same fields as the supervisor architecture. In addition, the *ReAct Agent* is closer to an end-to-end approach than to a pipeline approach, mitigating the error propagation risk by design.

Combining the ideas behind the supervisor and ReAct architecture, we propose the network architecture shown in Figure 3. This architecture combines the idea of a domain expert agent while widening the scope of each domain compared to the domain experts in the supervisor architecture. Therefore, the network architecture is built on an *Extraction Agent*, a *URI Mapping and Refinement Agent*, and a *Validation and Output Agent*.

The *Extraction Agent* combines the *Entity Extraction Agent* and *Triple Extraction Agent* from the supervisor architecture into a single agent. Our hypothesis is that LLMs can handle broader tasks as long as they remain within the same domain. Following that principle, there is a dedicated *URI Mapping and Refinement Agent*. This agent is also responsible for defining the search terms to be in the connected *URI Retrieval Tool* and for writing the results back in the appropriate field of the state.

Another domain is the creation of Turtle output, which is separate from extracting free-form triples or creating search terms. Building on that idea, we propose the *Validation and Output Agent* as a hallucination prevention agent in our network architecture. To accomplish its task, the *Validation and Output Agent* has access to the *Turtle-to-Label* and *Semantic Validation* tools.

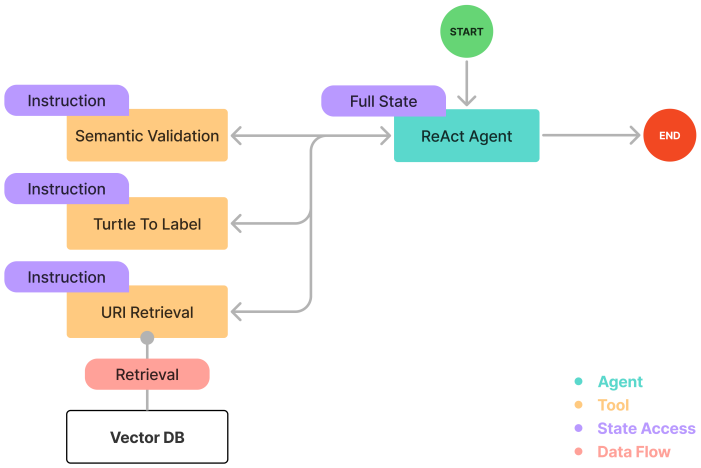

Figure 2: CIExMAS ReAct Architecture

To mitigate the risk of exceeding the context window and minimise prompt noise, the network architecture was designed to partition the state into domain-specific fields. Prior to agent execution, the text was saved in a single field within the domain. The extraction result of the *Extraction Agent* was saved within a `triple` field. The results of the *URI Mapping and Refinement Agent* were saved within a `URI mapping` field. Therefore, each result was present for each agent and could be updated by the respective agent.

To control agentic execution and prevent looping, we introduce the fields `last call`, `last response`, and `call trace`. In addition, we split the `instruction` field into `agent instruction` and `tool input`. All agents have read access to all fields of the shared state. The state design could narrow the context to its relevant information and highlight it without the need to provide the full message history.

The network architecture connects all agents. This enables agentic decision making. As such, the *Validation and Output Agent* could, for instance, hand off to the *Extraction Agent* or to the *URI Mapping and Refinement Agent*, depending on the type of error detected. If the *Validation and Output Agent*, for instance, detects a domain mismatch, it triggers a targeted back-propagation of formal errors to the *URI Mapping Agent* as corrective feedback. The *Extraction Agent* could route directly back to the *Validation and Output Agent* when its results have not changed, but it also has the ability to ask the *URI Mapping and Refinement Agent* to search URIs for newly extracted triples. This creates a highly flexible agentic system.

Ultimately, the supervisor and ReAct architecture share the same state and are the least complex. The network architecture is, on the one hand, more complex, as it keeps three fully autonomous agents, and, on the other hand, less complex, as it tries to keep resource use down through state design. In the end, the philosophy behind all architecture stays the same: creating a system that leverages the semantic understanding capabilities of a LLM.

4. Evaluation

In this section, we evaluate the performance of the proposed CIExMAS architectures. We first present the experimental setup, including the dataset and metrics, in Section 4.1. Subsequently, Section 4.2 presents a comparative analysis of our findings against state-of-the-art baselines, while Section 4.4 provides an in-depth interpretation of the results and highlights the remaining challenges in agentic cIE.

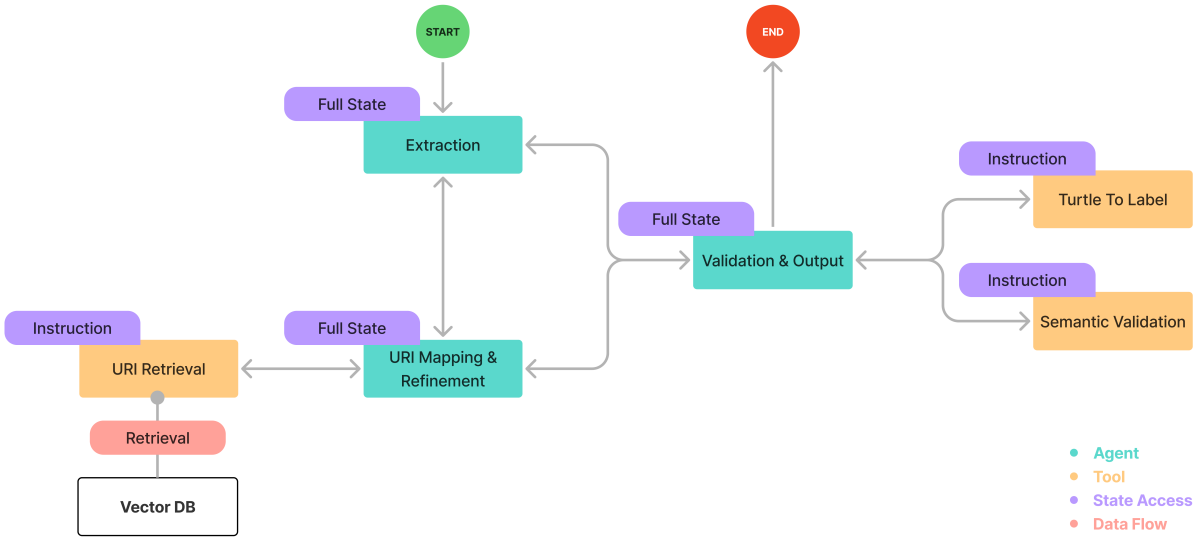

Figure 3: CIExMAS Network Architecture

4.1. Evaluation Setup

Common datasets for cIE, such as REBEL [15], tend to include skewness and low-quality triples [2]. Initial testing with REBEL on our approaches revealed low performance and unsolvable tasks for humans. As multiple LLM vendors recommend using LLMs for human-achievable tasks [20, 23], we decided to use Wiki-cIE [2] because of its reportedly high performance and human-solvable extractions.

The Wiki-cIE dataset consists of two generated datasets based on two models of the GPT-3.5 model series from OpenAI [2]. For the evaluation of CIExMAS, we leveraged the Wiki-cIE-text dataset, as it has shown higher performances and quality in our test. As the Wiki-cIE-text dataset is limited to a validation and test set, we decided to perform all engineering and optimisation on the validation set and evaluate on the test set. This also includes the examples used for in-context learning.

For evaluating, we used the Wiki-cIE-text test set and used only the first 50 examples sorted by ID. Therefore, we were able to easily reproduce the results across multiple setups and reduce errors while comparing our results to the comparison models. The choice of exactly 50 samples serves as a representative heuristic. While the specific number is arbitrary, it suits the purpose of CIExMAS to experiment with many different setups. Using the full split with 50,300 examples would have taken over 6 days, limiting the number of experiments we could have conducted. In addition, by comparing with other models evaluated on the full Wiki-cIE-text dataset, we can assess how representative our results are relative to the subset evaluation. Ultimately, we present both quantitative and qualitative analyses of our results.

For the final evaluation, the CIExMAS architectures used a vector database containing only entities and properties from the 50 Wiki-cIE-text samples. All CIExMAS architectures were based on Llama 3.3 70B. As a model provider for the results of this paper, we used Cerebras[2] and their unquantised hosting of Llama 3.3 70B. To ensure comparability across agent executions, the model temperature was set to zero, and a fixed seed was used for all experiments. All CIExMAS architectures were implemented using LangGraph[3]. $\text{GenIE}_{\text{T5-base}}$ and $\text{synthIE}_{\text{T5-large}}$ models of Josifoski et al. [10, 2] were used as reference models, as they had already been evaluated on Wiki-cIE-text and could be reused easily. Both were also evaluated on the same subset of Wiki-cIE that we used for CIExMAS.

The evaluation metrics were macro-averaged precision, recall and F_1 score following the proposal of Josifoski et al. [10]. As Josifoski et al. [10] evaluated only at triple level, we introduce separate metrics

[2]https://www.cerebras.ai
[3]https://www.langchain.com/langgraph

Table 1
Performance comparison of CIExMAS against state-of-the-art models on the synthIE dataset.

Model	Triples				
	Precision	Recall	F_1	Parental F_1	Related F_1
$\text{synthIE}_{\text{T5-large}}$	**0.835**	**0.832**	**0.831**	**0.841**	**0.858**
$\text{GenIE}_{\text{T5-base}}$	0.527	0.344	0.391	0.406	0.419
CIExMAS Variants					
Initial Supervisor Architecture	0.228	0.252	0.233	0.267	0.267
Supervisor Architecture	0.519	0.675	0.572	0.610	0.617
ReAct Architecture	0.674	0.517	0.572	0.616	0.626
Network Architecture	0.767	0.659	0.693	0.729	0.729

for subjects, objects, entities overall, and properties. This means, for instance, that whenever a subject is in both the predicted and test sets, it will count as a true positive. Whenever it is just in the predicted set, it will count as a false positive. The other way around is defined as a false negative.

In addition to the definition in [10], we propose a metric for soft matches and introduce it for both triples and properties. Soft matches occur whenever a property of a triple is either a sub- or superproperty of a property defined within the dataset. We define two types of soft matches: first, a parental soft match, if the extracted property is a subproperty of the target property. The parental soft match defines a match that, in theory, is correct, as the extracted property is narrower than what we expected. A knowledge graph reasoner would entail the knowledge graph with the same target property. Second, a related soft match, if the property is either a sub- or superproperty of the target property. We include the related soft match, as the difference between the cumulative related and narrowing parental soft matches indicates whether the tested model extracts general triples or matches the expected specificness. This allows further error investigation. To calculate those soft matches, we used hierarchy queries within a cycle-free dataset of Wikidata.

4.2. Results

In this section, we present the results of CIExMAS architectures compared to synthIE and GenIE. The section will start with our initial findings on the supervisor architecture in section 4.2.1, followed by the final results and a comparison with GenIE and synthIE in section 4.2.2[4]. In addition we outline the latency and cost used by each approach in section 4.3.

4.2.1. Initial Results and Shortcomings

The initial implementation was intended to serve as a baseline of the agentic system in cIE. As a consequence, our initial implementation of the supervisor architecture used simplified agents. Each agent used one-line prompts without examples. The initial implementation also did not back-propagate errors into the agents or use URI retrieval modes. In addition, the agents did not have to come up with correct Turtle strings as outputs; they were just required to produce semicolon-separated QNames of the triples. Later, we used scripts to transform those QNames into full URIs to compare the results with those of the other approaches.

This initial implementation did not achieve the GenIE model's F_1 score by a margin of 0.158. Nevertheless, it already showed the strong capability of the agentic systems to extract subjects, objects and entities. In those scores, it either matches or beats GenIE. Nevertheless, it falls short in detecting the correct properties, which ultimately leads to a lower F_1 score than GenIE.

Nevertheless, the initial supervisor architecture showed potential, but it had significant flaws in its behaviour. For instance the output was formatted in an invalid format and therefore could not be evaluated. The malformed output was detected in 12% of the outputs.

[4]More detailed results are available at https://github.com/maxlautenbach/CIExMAS

Table 2
Comparison of F_1 scores by category for CIExMAS and state-of-the-art models on the Wiki-cIE dataset.

Model	Category F_1				Property Hierar.	
	Subjects	Objects	Entities	Properties	Parental F_1	Related F_1
$\text{SynthIE}_{\text{T5-large}}$	**0.929**	**0.921**	**0.933**	**0.882**	**0.894**	**0.912**
$\text{GenIE}_{\text{T5-base}}$	0.689	0.583	0.712	0.459	0.501	0.527
CIExMAS Variants						
Initial Supervisor Arch.	0.393	0.448	0.460	0.382	0.498	0.506
Supervisor Architecture	0.810	0.870	**0.933**	0.656	0.739	0.746
ReAct Architecture	0.910	0.759	0.846	0.658	0.727	0.746
Network Architecture	0.905	0.854	0.897	0.735	0.790	0.800

In various traces, we observed that the *Supervisor Agent* either looped over the same agent with the same input or skipped the *URI Retrieval Tool* and hallucinated QNames instead of retrieving them. In most traces, it also did not iterate over incorrect results. Instead, it reused the *URI Retrieval Tool* with different inputs; it reused it with the same input. In addition, it produced malformed URI retrieval inputs that did not conform to the requested XML tag format. In such a case, the *URI Retrieval Tool* will search using an empty input in this implementation. These observed shortcomings in the initial version directly informed the implementation of the self-repair mechanisms described in Section 3.2.

Ultimately, this initial implementation was already close to GenIE, while still carrying major flaws that could be resolved in subsequent iterations. Therefore, the initial results already showed shortcomings but also paved the way for architectures such as the ReAct architecture and network architectures.

4.2.2. Final Results

In this section, we will present the results of the CIExMAS architectures compared to $\text{synthIE}_{\text{T5-large}}$ and $\text{GenIE}_{\text{T5-base}}$ on a subset of Wiki-cIE data. We will compare their triple F_1 scores and category F_1 scores.

The network architecture achieved a triple F_1 score of 0.693, which is 0.460 higher than the initial supervisor architecture, but exceeded by synthIE by another 0.138. Within the group of CIExMAS variants, the supervisor and ReAct architecture share the same triple F_1 score, which is 0.121 lower than the network architecture. All CIExMAS variants, except for the initial supervisor architecture, could achieve an error rate of 0% and outperform the results of the $\text{GenIE}_{\text{T5-base}}$ model across all metrics.

This result includes a direct comparison of a very simplistic initial version of the supervisor architecture against a developed version that includes error handling, prompt engineering, and filtering modes. In this comparison, the developed version has a 0.339 higher triple F_1 score. In addition, all other categories' F_1 scores consistently outperformed the initial supervisor architecture.

Both the supervisor and the ReAct architecture share the same triple F_1, whereas the architecture and, especially, the tools connected to it differ. The ReAct architecture, with its single agent and enhanced validation tooling, produces more triples that are actually expected by the dataset, whereas the supervisor architecture, with its multiple agents for extraction and URI mapping, detects more expected triples overall.

The network architecture surpasses the high precision of the ReAct architecture, while keeping the higher recall of the supervisor architecture. This makes the network architecture the best overall CIExMAS architecture. Nevertheless, the network and ReAct architecture showed low levels of iteration in the traces. The only architecture with a higher iterative behaviour was the supervisor architecture. The traces showed that in 40% of them, the *URI Mapping Agent* or the *Triple Extraction Agent* was called twice or more.

When examining the parental and related triple F_1 scores, it is apparent that the CIExMAS approaches exhibit larger gaps between these scores and the standard triple F_1 than the comparison models do. For instance, the parental triple F_1 of $\text{synthIE}_{\text{T5-large}}$ is only 0.010 higher than the standard. The final

Table 3
Resource consumption and latency comparison of CIExMAS architectures for 50 samples on Wiki-cIE-text.

Architecture	Duration	Input Tokens	Output Tokens	Total Tokens	Cost ($)
Initial Supervisor	13m 29s	439,096	17,131	456,227	0.3938
Supervisor	9m 00s	743,103	88,838	831,941	0.7382
ReAct	9m 02s	600,656	50,590	651,246	0.5713
Network	11m 48s	687,571	103,919	791,490	0.7091

CIExMAS approaches exhibit a gap of 0.036-0.044 between the standard F_1 and the parental F_1. In addition, most of this gap is explained by the CIExMAS approaches 'extraction properties, which are superproperties of the expected ones.

When focusing on the categories, it becomes apparent that the CIExMAS approaches come very close to the best-performing comparison model, synthIE$_{\text{T5-large}}$. Regarding subject extraction, the F_1 score of synthIE$_{\text{T5-large}}$ is 0.024 higher than that of the network architecture and 0.019 higher than the ReAct architecture. The gap between synthIE$_{\text{T5-large}}$ and the network architecture regarding objects is 0.067 and regarding entities 0.036. It should be noted that the supervisor architecture matches the F_1 of synthIE$_{\text{T5-large}}$ for entity extraction without the usage of any additional validation tools.

Whereas subject, object, and entity extraction overall reveal only small gaps, property extraction shows otherwise. The best architecture in the CIExMAS group remains the network architecture. It shows a 0.077-0.079 higher F_1 score than the ReAct and supervisor architectures, respectively. Nevertheless, the synthIE$_{\text{T5-large}}$ model surpasses the property F_1 by a margin of 0.147.

The same problem of detecting super-properties is also apparent in the CIExMAS approaches for the parental and related property F_1 scores. The initial supervisor architecture shows a 0.116 higher parental property F_1 than the strict one. The remaining approaches show a gap of 0.055-0.083. The gap of synthIE$_{\text{T5-large}}$ was again only 0.012.

All in all the results of the CIExMAS approaches stayed below the results of synthIE$_{\text{T5-large}}$, while outperforming GenIE$_{\text{T5-base}}$. The CIExMAS approaches could nearly match synthIE$_{\text{T5-large}}$ regarding subject, object and entity extraction. The gap between synthIE$_{\text{T5-large}}$ and the CIExMAS models is primarily the property extraction.

4.3. Latency and Cost

Following up on the performance results, we present the latency and cost comparisons for the CIExMAS approaches in table 3. The cost was calculated based on the prices while evaluating the approaches, which were 0.85$ for one million input tokens and 1.25$ for one million output tokens[5].

The ReAct architecture, as the simplest, consumes the fewest tokens while being as time-expensive as the supervisor architecture. The network architecture ran 2 minutes slower on the test dataset. As outlined earlier, the initial supervisor exhibited inefficient loops, resulting in the highest latency of over 13 minutes during the test run.

Tokenwise, the network architecture can show the outlined token-efficient design. Compared with the supervisor and ReAct approaches, the network architecture consumes the fewest input tokens. Nevertheless, it consumes more output tokens than any other tested architecture, as it also requires more complex outputs to comply with the state.

In total, the network architecture consumes nearly as many tokens as the supervisor architecture. Therefore, both executions were similarly expensive, at 71 cents and 74 cents, respectively. The React architecture execution was considerably cheaper, at 57 cents. Overall, the simplest architecture is the cheapest to execute but as fast as more complex ones. The cheapest to execute is the one-line system prompt, which later results in longer inference time.

[5]Pricing as of 17. January 2026 available under https://web.archive.org/web/20260117025315/https://www.cerebras.ai/pricing

4.4. Discussion

The final results of the comparison between the CIExMAS approaches, GenIE, and synthIE reveal two main findings. First, fine-tuned models like synthIE perform significantly better than general approaches like the network architecture in CIExMAS. Secondly, the lower F_1 scores in CIExMAS models can mainly be traced back to the property extraction performance of the CIExMAS models. Therefore, we have identified property extraction as the main shortcoming of the CIExMAS models.

When going into detail, each architecture has its strengths. First of all, the supervisor architecture could be improved by implementing error handling, prompt engineering, and an enhanced filtering mode for URIs. In combination with the *Entity Extraction Agent*, this yielded the same entity F_1 scores as $\text{synthIE}_{\text{T5-large}}$. This already shows that pre-trained LLMs matches the entity extraction performance of fine-tuned Transformers with low effort.

In contrast, we used the ReAct architecture to assess a single agent's capability in the cIE domain. In combination with the validation tools, the ReAct architecture matched the results of the supervisor architecture. It showed that using a single agent reduces the number of extracted triples while improving their quality. Nevertheless, this effect was amplified by the validation tools used with the ReAct architecture. Those tools prevented the agent from outputting semantically incorrect triples.

The combination of both ideas, the ability of a single agent to extract correct triples and the idea of a clear and narrow context for each agent worked out the best within the network architecture. Nevertheless, the network architecture, with its design for iteration, has not often shown them. Based on our findings, this was mainly due to the agents' confidence in a correct result, but also to the prompt design, as we needed to balance between looping and a straight, pipeline-like execution. This implies that the setup we used couldn't utilise the full potential of agentic behaviour.

All of our architectures fell short in their ability to extract properties. This effect can be explained to some extent, as our approach tends to greater specificity in the chosen property. Therefore, the extracted and the target property stay related. Nevertheless, even on this soft-match metric, our approaches were outperformed by $\text{synthIE}_{\text{T5-large}}$. This shows that the CIExMAS approaches can only detect a high fraction of the target properties, but not all of them.

Our evaluation shows that even with a few iteration steps, a pre-trained LLM can solve the cIE task to a great extent when used in an agentic setup. Especially when compared to highly trained models like $\text{synthIE}_{\text{T5-large}}$ and $\text{GenIE}_{\text{T5-base}}$, it demonstrates the high potential of LLM-based agents in this domain. In addition, we did not limit the set of tokens the LLMs could generate, unlike Josifoski et al. [2].

This paper used a representative mid-sized LLM, Llama 3.3 70B, as it was an accessible, open-source model. As of today, Llama 3.3 70B has been surpassed by over 150 other models on the LMSYS Chatbot Arena leaderboard[6], including multiple open-source models with a similar number of parameters. Additionally, we self-assigned a context limit of 8192 tokens, which required us to use a compact agentic state by design. It must be assumed that more recent, especially proprietary, LLMs will surpass the performance achieved with Llama 3.3 70B.

Another approach that limited the potential of CIExMAS models is iterative manual prompt engineering. Recent methods such as MIPRO [24] and ACE [25] have shown promising results in prompt optimisation. Nevertheless, we proposed a generalisable approach for cIE, which we might undermine by using prompt-based techniques, as they could learn to perform well on a single dataset.

As we used LLMs, the computation time per evaluation on Cerebras exceeded 9 minutes. This would have led to a total time of approximately 6 days for all 50.300 Wiki-cIE test examples. A sequential execution was needed due to Cerebras's rate limits. When using other providers like DeepInfra.com, execution time would be approximately 10 times higher, while rate limits would be lower, leading to the same problem. In addition, the cost would have been at its lowest with approximately 570 $ for the ReAct architecture and 740$ for the Supervisor architecture. As a consequence, we decided to use 50 samples to keep the iteration speed high and execution costs low, and to test and evolve other unmentioned architectures. In addition, the use of Cerebras accelerated the process, but at a price, which is another reason why we were limiting the evaluation to 50 samples.

[6]Available under https://arena.ai/leaderboard/text

In the end, the low number of sample limits the significance of the results presented in the paper. Nevertheless, the potential and caveats of an LLM-based multi-agent system in the cIE domain can be illustrated. Additionally, the results of $\text{synthIE}_{\text{T5-large}}$ and $\text{GenIE}_{\text{T5-base}}$ achieved on the full dataset are 0,1 F_1 higher for synthIE and 0,25 lower for GenIE compared to our subset. Despite these slight variations, the consistency in performance trends indicates that our subset is a good exploratory proxy. This justifies the smaller sample size as a balanced trade-off between statistical significance and computational feasibility.

A direct comparison with other cIE approaches, such as DISCIE [12], was omitted, as this would have required significant effort to fine-tune these models on the Wiki-cIE dataset. While it is theoretically possible to use other models on the Wiki-cIE dataset, I would have missed the exploratory focus of this work.

In the final evaluation, we indexed the expected entities and properties. Therefore, we could expect the performance to be higher than in a real-world test, where all Wikidata entities and properties would be indexed. With more similar entries expected in the vector database, the agent's ability to disambiguate between similar entities and properties based on the context of the given text is expected to be more influential on the overall outcome. Other than that, the performance could also have been increased in terms of parental and related F_1-scores, as more super and sub-properties would have been available.

In conclusion, CIExMAS is a proof point for using LLMs without fine-tuning in the domain of cIE by leveraging multi-agent systems. The results show that the gap between fine-tuned cIE models and CIExMAS approaches is 0.138 in triple F_1, while the CIExMAS approach has seen very few examples in the Wiki-cIE dataset. Additionally, we have shown that agentic systems can match fine-tuned cIE models on entity extraction. All results have been achieved by following the general architecture of CIExMAS, which allows it to be adapted to different knowledge bases later on, whereas fine-tuned cIE models needed retraining. Ultimately, the potential of a generalist system like CIExMAS is high, as ideas such as automatic prompt engineering and more advanced language models can be added in future work.

5. Conclusion and Future Work

In this paper, we present CIExMAS, a collection of LLM-based multi-agent systems for cIE. Our approach was to leverage pre-trained LLMs in an agentic manner, without pre-training, to extract triples from text. All CIExMAS approaches relied solely on LLMs and Tools for mapping URIs, as well as on validation tools. In our approach, we present three architectures, the network, the ReAct and the supervisor architecture and compare them on a subset of the Wiki-cIE dataset against the best synthIE and GenIE models. In our evaluation, we use beyond-standard cIE metrics, utilising parental and related F_1 scores, to gain insights into close misses in predicting correct triples.

Our work shows multiple findings. (1) LLMs are able to extract triples out of text with their corresponding URIs when used in a multi-agent system utilising grounding mechanisms. Nevertheless, (2) fine-tuned and highly specialised models like synthIE can surpass our CIExMAS approaches by approximately 0.138 in triple F_1. (3) We could reach the 0.138 gap in triple F_1, while using a representative mid-sized LLM in an exploratory setup without the focus on optimising. This work reveals a starting point for further exploration in the area of multi-agent system for cIE. Approaches similar to CIExMAS can be adapted quickly to various knowledge graphs and domains. Without the need for intensive re-training, such general approaches are opening up for real-world usage.

Our findings explicitly pinpoint property extraction as the primary remaining frontier for LLM-based agentic cIE. To address this, future work could enhance depth in optimisation using automatic optimisation approaches such as MIPRO [24] or ACE [25]. Furthermore, to resolve the second limitation of LLM inference cost, distilling bigger model outputs to smaller models, or especially fine-tuning smaller models, could be effective.

Similar work in the field of oIE has already shown that smaller models can surpass larger ones when

fine-tuned. Approaching the cost of CIExMAS and the fast evolution cycles in the LLM research, parts of the tasks done by the agents of CIExMAS could be done by specialised and fine-tuned small language models. In combination with automated prompt optimisation, such an approach could be trained quickly while also reducing inference costs and increasing inference speed. This is a possible solution for the performance gap in property extraction.

Domain-wise, the cIE approach does not, in our opinion, depict an end-to-end real-world use case. The extraction of triples from text, which can later be mapped to predefined entities and properties in a knowledge graph, still leaves the gap in the need to define those entities and properties. In future work, we propose closing this gap by enabling the agentic system to create entities and properties as needed, enabling agentic ontology evolution. In this case, the text-to-knowledge graph will be usable within various domains. In summary, we think that participating in the rapid evolution of agentic systems and LLM development in the domain of text-to-knowledge graphs is crucial to providing agents with a semantically rich knowledge base.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly and Gemini 3 Thinking-Mode in order to: Grammar and spelling check. Further, the authors used Gemini 3 Thinking-Mode in order to: Peer review simulation. After using these tools/services, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] M. Korolov, Knowledge graphs: the missing link in enterprise ai, CIO (2025). URL: https://www.cio.com/article/3808569/knowledge-graphs-the-missing-link-in-enterprise-ai.html, contributing writer.

[2] M. Josifoski, M. Sakota, M. Peyrard, R. West, Exploiting asymmetry for synthetic training data generation: Synthie and the case of information extraction, 2023. URL: https://arxiv.org/abs/2303.04132. arXiv:2303.04132.

[3] L. Xue, D. Zhang, Y. Dong, J. Tang, Autore: Document-level relation extraction with large language models, CoRR abs/2403.14888 (2024). doi:10.48550/ARXIV.2403.14888. arXiv:2403.14888.

[4] Z. Chen, Z. Li, Y. Zeng, C. Zhang, H. Ma, Gap: A novel generative context-aware prompt-tuning method for relation extraction, Expert Systems with Applications 248 (2024) 123478. doi:10.1016/j.eswa.2024.123478.

[5] Y. Shi, G. Jiang, T. Qiu, D. Yang, Agentre: An agent-based framework for navigating complex information landscapes in relation extraction, 2024. URL: https://arxiv.org/abs/2409.01854. arXiv:2409.01854.

[6] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, T. Scialom, Gaia: a benchmark for general ai assistants, 2023. URL: https://arxiv.org/abs/2311.12983. arXiv:2311.12983.

[7] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, X. Zhang, Large language model based multi-agents: A survey of progress and challenges, 2024. URL: https://arxiv.org/abs/2402.01680. arXiv:2402.01680.

[8] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang, T. Gui, The rise and potential of large language model based agents: A survey, 2023. URL: https://arxiv.org/abs/2309.07864. arXiv:2309.07864.

[9] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, P. Clark, Self-refine: Iterative refinement with self-feedback, 2023. URL: https://arxiv.org/abs/2303.17651. arXiv:2303.17651.

[10] M. Josifoski, N. De Cao, M. Peyrard, F. Petroni, R. West, Genie: Generative information extraction, 2021. URL: https://arxiv.org/abs/2112.08340. arXiv:2112.08340.

[11] Q. Guo, Z. Jin, X. Qiu, W. Zhang, D. Wipf, Z. Zhang, Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training, 2020. URL: https://arxiv.org/abs/2006.04702. arXiv:2006.04702.

[12] C. Möller, R. Usbeck, Discie–discriminative closed information extraction, in: Lecture Notes in Computer Science, Springer Nature Switzerland, 2024, pp. 23–40. doi:10.1007/978-3-031-77850-6_2.

[13] B. D. Trisedya, G. Weikum, J. Qi, R. Zhang, Neural relation extraction for knowledge base enrichment, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2019. URL: http://dx.doi.org/10.18653/v1/P19-1023. doi:10.18653/v1/p19-1023.

[14] F. Mesquita, M. Cannaviccio, J. Schmidek, P. Mirza, D. Barbosa, KnowledgeNet: A benchmark dataset for knowledge base population, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 749–758. URL: https://aclanthology.org/D19-1069/. doi:10.18653/v1/D19-1069.

[15] P.-L. Huguet Cabot, R. Navigli, REBEL: Relation extraction by end-to-end language generation, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 2370–2381. URL: https://aclanthology.org/2021.findings-emnlp.204/. doi:10.18653/v1/2021.findings-emnlp.204.

[16] Y. Liu, T. Zhang, Z. Liang, H. Ji, D. L. McGuinness, Seq2rdf: An end-to-end application for deriving triples from natural language text, 2018. URL: https://arxiv.org/abs/1807.01763. arXiv:1807.01763.

[17] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, P. Resnik, The prompt report: A systematic survey of prompt engineering techniques, 2025. URL: https://arxiv.org/abs/2406.06608. arXiv:2406.06608.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, 2020. URL: https://arxiv.org/abs/2005.14165. arXiv:2005.14165.

[19] J. Hadfield, B. Zhang, K. Lien, F. Scholz, J. Fox, D. Ford, How we built our multi-agent research system, https://www.anthropic.com/engineering/built-multi-agent-research-system, 2025. Accessed: 2025-06-17.

[20] Anthropic, Building effective agents, 2024. URL: https://www.anthropic.com/engineering/building-effective-agents, accessed: 2025-06-03.

[21] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 2318–2335. URL: https://aclanthology.org/2024.findings-acl.137/. doi:10.18653/v1/2024.findings-acl.137.

[22] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, 2023. URL: https://arxiv.org/abs/2210.03629. arXiv:2210.03629.

[23] OpenAI, A practical guide to building agents, https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf, 2025. Whitepaper, OpenAI.

[24] K. Opsahl-Ong, M. J. Ryan, J. Purtell, D. Broman, C. Potts, M. Zaharia, O. Khattab, Optimizing

instructions and demonstrations for multi-stage language model programs, 2024. URL: https://arxiv.org/abs/2406.11695. arXiv:2406.11695.

[25] Q. Zhang, C. Hu, S. Upasani, B. Ma, F. Hong, V. Kamanuru, J. Rainton, C. Wu, M. Ji, H. Li, U. Thakker, J. Zou, K. Olukotun, Agentic context engineering: Evolving contexts for self-improving language models, 2026. URL: https://arxiv.org/abs/2510.04618. arXiv:2510.04618.