

UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Rasul Nabiyev

Collaborative Multi-Agent Architecture for Domain-Agnostic Named Entity Recognition

Master's Thesis (30 ECTS)

Supervisor(s): Hendrik Šuvalov, MSc
Karl-Oskar Masing, BSc

Tartu 2025

Collaborative Multi-Agent Architecture for Domain-Agnostic Named Entity Recognition

Abstract:

Named Entity Recognition (NER) traditionally requires extensive domain-specific training data to achieve satisfactory performance for a given domain. Recent advancements in large language models have enabled the development of NER systems without supervised training, though this approach still requires careful prompt engineering and may need external knowledge augmentation during inference. This thesis introduces a novel domain-agnostic NER framework based on a collaborative multi-agent architecture that can adapt to any domain given only entity definitions and their descriptions. The framework consists of 4 high-level components: a team of agents, a metaprompter, a chat supervisor and a grounding engine. The system requires no training data or prompt engineering for new domains, operating as a few-shot solution for NER tasks. The framework's performance is evaluated across 4 distinct domains using standard NER benchmark datasets. Our evaluation shows that the multi-agent approach outperforms the baseline of few-shot NER with single LLM call in 3 out of 4 benchmarks, suggesting a promising direction for domain-agnostic NER. Ablation studies demonstrate varying effectiveness of each component on the system's performance depending on the domain, with the combination of three specialized agents and grounding engine proving generally most effective in all tested domains.

Keywords: named entity recognition, multi-agent systems, large language models

CERCS: P176 - Artificial intelligence; P170 - Computer science, numerical analysis, systems, control

Mitmikagendilise arhitektuuri rakendamine nimeolemite märgendamiseks

Lühikokkuvõte:

Nimiolemite märgendamine (NER) nõuab mingis valdkonnas rahuldava tulemuse saamiseks tüüpiliselt massiivset valdkonnapõhist treeningandmestikku. Viimased edusammud suurte keelmodelite vallas on võimaldanud välja töötada NER-süsteeme juhendatud õppimiseta. See lähenemine nõuab aga hoolikat viipade loomist (*prompt engineering*) ja võib vajada lisaks domeenieksperti sisendit. Käesoleva lõputöö raames valmis uus domeenist sõltumatu NER-raamistik, mis põhineb mitme agendi koostöö arhitektuuril ning suudab kohaneda erinevate valdkondadega, vajades vaid olemite definitsioone ja kirjeldusi. Raamistik koosneb neljast kõrgetasemelisest komponendist: agentide meeskonnast, metaviipajast, vestluse juhendajast ja tõendajast. Süsteem ei vaja uutes valdkondades olemeid tuvastades treeningandmeid ega eraldi viiba kohandamist, talitades nõnda väheste näidete lahendusena (*few-shot solution*) NER ülesannete raames. Raamistiku jõudlust hinnatakse neljas erinevas valdkonnas, kasutades standardseid NER-süsteemide hindamisandmeid. Saadud tulemused toetavad domeenist sõltumatu NER-süsteemi loomise võimalikkust, saavutades väheste näidetega üksikust viipamisest parema tulemuse kolmes valdkonnas neljast. Abaltsiooniuringud näitavad, et iga komponendi efektiivsus varieerub valdkonnati, kusjuures kolme agendi ja tõendaja kombinatsioon osutus valdkondade üleselt üldiselt kõige tõhusamaks.

Võtmesõnad: Nimeüksuste märgendamine, mitmeagendilised süsteemid, suured keelmodelid

CERCS: P176 - Tehisintellekt; P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1	Introduction	6
2	Background and related work	7
2.1	Transformers, Large Language Models, and few-shot learning	7
2.2	Task transformation and NER using few shot learning	8
2.3	Agentic design	10
2.3.1	Tool use	10
2.3.2	Chain of Thought	11
2.3.3	ReAct	11
2.4	Retrieval Augmented Generation	11
3	Methodology	12
3.1	AgenticNER	13
3.1.1	Metaprompter	14
3.1.2	The agent team	16
3.1.3	Chat supervisor	17
3.1.4	Grounding engine	19
3.2	Evaluation	19
3.2.1	Evaluation metrics	19
3.2.2	Challenges with evaluation	21
3.2.3	Datasets	23
3.3	Baseline	24
3.4	Experiment setup	26
3.4.1	Large Language Models	26
3.4.2	Example sampling	27
3.4.3	Knowledge Base	28
3.4.4	Temperature and non-determinism	28
4	Results	29
5	Ablation study	31
5.1	Variants	31
5.2	Impact on Precision	31
5.3	Impact on Recall	33
5.4	Impact on F1 score	34
6	Limitations and future work	36
6.1	Improving recall	36
6.2	More robust evaluation methods for generative NER	36

6.3 Better knowledge base integration	37
7 Conclusion	37
References	42
Appendix	43
I. Source code	43
II. Example conversation between agents in AgenticNER	43
II. Licence	47

1 Introduction

Named Entity Recognition(NER) is a task in Natural Language Processing(NLP), where a machine learning model or software system is required to locate and classify named entities in unstructured text. A named entity can be a word or phrase in the text being analyzed that corresponds to an object in the real world such as a person, or location, or an abstract concept such as revenue, date, or time. The named entities of interest can be different in different domains. For example, in microbiology, we might be interested in identifying proteins, genes, cell types, whereas in astronomy the entities would be astronomical objects or units. NER has numerous practical applications including but not limited to information retrieval[1], text mining[2, 3], knowledge graph construction[4, 5, 6], and machine translation[7, 8].

The methods used for NER have evolved significantly over the years. Early NER systems utilized rule-based approaches that used hand-crafted rules. Introduction of machine learning techniques allowed more flexible NER systems to be developed and NER evolved further with utilization of deep learning. After introduction of Transformer architecture[9] NER performance reached new heights thanks to several breakthroughs in the field[10].

Large Language Models(LLMs) - the latest advancement in Artificial Intelligence and NLP - introduced a new paradigm to the field called in-context learning[11]. In-context learning, often used interchangeably with few-shot or zero-shot learning, is the capability of LLMs where models can apply their general pattern recognition skills learned during unsupervised pre-training to new tasks they haven't been explicitly trained or fine-tuned on. Several methods have been proposed and studied on using in-context learning for NER task such as GPT-NER[12] and PromptNER[13] demonstrating comparable performances to fully supervised baselines on widely adopted NER benchmarks. Even though LLMs perform well on standard NER benchmarks they face challenges in more nuanced scenarios, especially when the entity types are more granular and domain-specific. These challenges become increasingly apparent in tasks that require a deep understanding of a specific domain, where generic language models cannot effectively grasp the subtleties of the domain without targeted training or fine-tuning. This limitation necessitates specialized adaptations for each domain or hybrid data-augmented systems to compensate for the lack of domain understanding of off-the-shelf LLMs[10].

As reasoning capabilities of LLMs improved, Agentic design has emerged as a new trend in AI research. Compared to prompting an LLM to generate a final answer token by token, agentic workflows allow an LLM to iterate over its work, use tools to interact with and receive feedback from the external world, and decide when to stop to give a final answer to fulfill a given task. Multi-agent architecture is a version of agentic design where instead of a single LLM agent multiple LLM agents are employed to perform a complex task in a collaborative setting. Both single and multi-agent workflows improve performance on complex tasks such as code generation compared to the baseline of

simpler few-shot prompting approach[14, 15].

While in-context learning for NER and Agentic design for complex tasks have extensive coverage in scientific literature[10, 16] the usage of multi-agent architecture for NER is largely under-explored. In this thesis, we introduce AgenticNER, a framework for domain-agnostic NER based on few-shot learning and collaborative multi-agent design. The framework requires no training data or prompt engineering for new domains, providing a few-shot solution for NER tasks in any arbitrary domain. We evaluate the performance of our approach with a baseline of few-shot single LLM call method on benchmarks across 4 different domains. We perform an extensive ablation study and discuss the contribution of each component of AgenticNER on benchmark results. Finally, we discuss the limitations of our approach and discuss future work to overcome those limitations.

Claude AI assistant¹ has been used while writing this thesis to fix grammar errors, rephrase, and break-down complex sentences(author's) to improve readability.

2 Background and related work

This section provides an overview of the key research and developments in the field that are relevant to this thesis. We outline the important techniques, methods, and recent advancements that have laid the groundwork for the current study and made the implementation of AgenticNER possible.

2.1 Transformers, Large Language Models, and few-shot learning

The transformer architecture, introduced in the paper titled "Attention is All You Need"[9], has revolutionized NLP and deep learning fields by providing a more efficient alternative to the widely used recurrent neural networks(RNN), specifically, a version of RNNs called LSTM[17]. The new architecture used a self-attention mechanism that allows for parallel processing and more effective handling of long-range dependencies in sequential data. Unlike recurrent networks that process input sequentially, transformers use multi-head attention to simultaneously weigh the importance of different parts of the input sequence, enabling a more nuanced understanding of context and relationships within the text. This architecture became foundational in NLP, allowing for significant advances in machine translation, text generation, and language understanding through models like BERT[18] and GPT[19].

Thanks to the efficient transformer architecture which allowed faster training of large deep-learning models a new research direction emerged which studied characteristics of models based on transformer architecture as they scale. Kaplan et al.[20] revealed a

¹<https://claude.ai/>

consistent relationship between model performance and key factors such as model size, dataset size, and computational resources. The scaling laws in this study demonstrated that language model performance improves predictably as these factors scale up, observable across several orders of magnitude. The takeaway was that larger models exhibit significantly higher sample efficiency, suggesting that optimal compute-efficient training involves using very large models with relatively modest amounts of data. This finding led to the development of GPT-3, the largest language model to date, and the concept of in-context learning was introduced in the paper titled "Language Models are Few-Shot Learners"[11]. In this research creators of GPT-3 showed that by presenting the model with just a few examples or even no examples at all LLMs could perform complex tasks across multiple domains, from translation to mathematical reasoning, without traditional task-specific fine-tuning. The study suggested a more generalized approach to NLP and more broadly to artificial intelligence by revealing the adaptive capabilities of massive language models.

2.2 Task transformation and NER using few shot learning

Conventionally NER is a sequence labeling or sequence classification task. Given a sequence T of N tokens $T = (t_1, t_2, t_3, \dots, t_n)$ the task is creating a set of tuples (I_b, I_e, c) where b and e correspond to the beginning and ending indices of the named entity respectively and c is the category of the entity which is selected from a predefined set of k categories $C = (c_1, c_2, \dots, c_k)$ [10]. Here both b and e should be within the interval of $[1, N]$. For example, in the sentence "Barack Obama was born in Honolulu.", a NER system would label "Barack Obama" as a person and "Honolulu" as a location as illustrated in Figure 1.

To use in-context learning and reasoning capabilities of generative large language models we have to formulate NER as a generation task rather than sequence labeling or sequence classification problem. GPT-NER[12], one of the earlier works on the application of LLMs to NER task, proposes feeding the input sequence $T = (t_1, t_2, \dots, t_n)$ to a generative language model alongside prompt P and examples of input-output sequence pairs (E, E') in order for LLM to generate an output sequence of $T' = (t'_1, t'_2, \dots, t'_n)$ where t'_x is expected to be either the original token t_x from the input sequence or a new token with an added special marker indicating the token t_x belonging to the entity category c of interest. If there are k different categories, the process is repeated for each category with a separate category-specific prompt, since the output sequence T' only indicates whether the original tokens in sequence T match a specific category, but does not specify which category they belong to. Figure 2 demonstrates an example of labeling location entities in the input sequence using a generative large language model. PromptNER[13] proposes a different approach where an LLM is given entity types and their definitions as input and is prompted to list entities from the input sequence alongside the explanation of why the entity is labeled as a certain entity type.

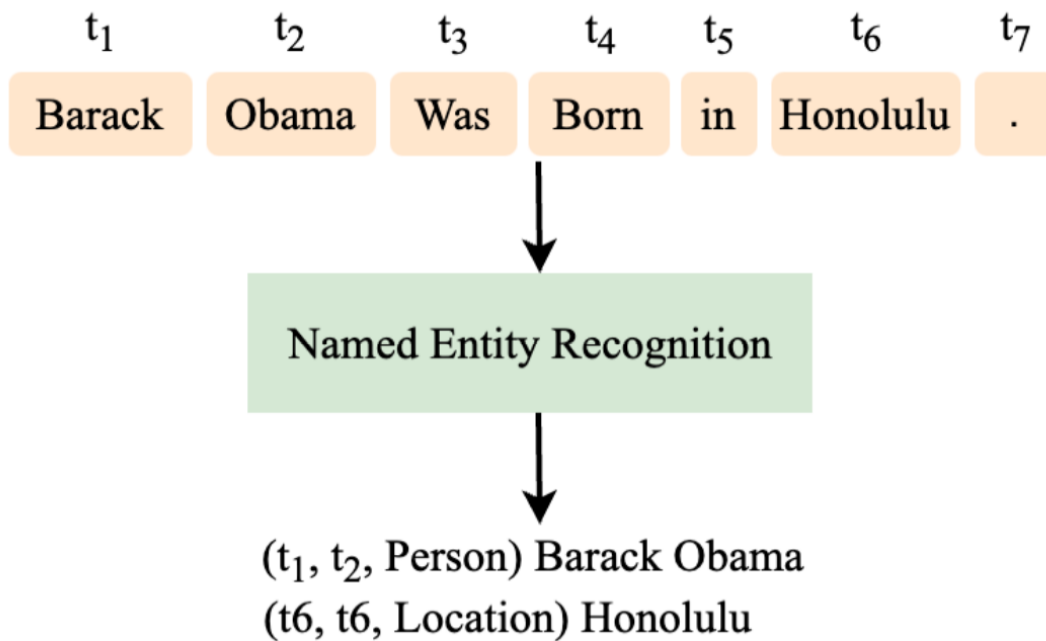


Figure 1. For a sequence of input tokens a NER system outputs boundaries of named entities and their category[10]

This method leverages a technique called Chain of Thought[21] which helps to improve reasoning capabilities of LLMs by prompting them to generate intermediate reasoning steps while solving a complex task.

I am an excellent linguist. The task is to label location entities in the given sentence. Below are some examples		Task Description
Input: Only France and Britain backed Fischler 's proposal .	Example 1	Few-shot Demonstrations
Output: Only @@France## and @@Britain## backed Fischler 's proposal .		
Input: Germany imported 47,600 sheep from Britain last year , nearly half of total imports .	Example 2	
Output: @@Germany## imported 47,600 sheep from @@Britain## last year , nearly half of total imports .		
Input: It brought in 4275 tonnes of British mutton . some 10 percent of overall imports .	Example 3	
Output: It brought in 4275 tonnes of British mutton . some 10 percent of overall imports .		
Input: China says Taiwan spoils atmosphere for talks .		Input Sentence
Output: @@China## says @@Taiwan## spoils atmosphere for talks .		

Figure 2. Example prompt, few-shot examples, input and output sequences from GPT-NER[12]

2.3 Agentic design

Autonomous AI agents has been an important area of research in academia and industry for a long time. AI agents or Agentic AI systems are defined by their ability to take actions on and interact with their environments in order to achieve a goal set by their user over an extended timeframe, without requiring explicit, pre-programmed instructions for each behavior. By this definition, Agentic AI is different than other general-purpose AI systems such as LLMs that can only complete a limited set of real-world tasks when interaction with them is constrained to a chat UI or API interface. However, the line between general-purpose LLMs and AI agents is not clear since both AI systems can complete certain complex tasks over an extended period of time² without predefined behavior. Shavit et al.[22] defines "agenticness" of AI system as *"the degree to which a system can adaptably achieve complex goals in complex environments with limited direct supervision"* and breaks down the concept of "agenticness" into several components such as goal complexity, environmental complexity, adaptability, and independent execution. In this thesis, we adhere to this definition of Agentic AI.

Recent advancements in the reasoning capabilities of large language models caused increased interest in LLM-based agents. LLM based AI agents can perform diverse tasks such as code generation, scientific research, creative writing, customer support by interacting with tools and resources provided by user. In these systems, the LLM is integrated into a direct feedback loop with an external environment and resources, such as the internet, databases, ticketing systems, or command line. This integration enables the LLM to select and execute actions on the available resources in order to fulfill a complex task[23]. A multi-agent system is an evolution of the single-agent system where the emphasis is on assigning a different role to each agent and allowing the agents to interact with each other on top of interacting with their environments. This setup aims to utilize the collaborative decision-making and problem-solving processes of the agents for a common objective, similar to human group dynamics in similar scenarios.

To build an effective multi-agent AI system designing an effective agentic architecture is crucial. If we compare Agentic AI systems to traditional deep learning designing the agentic architecture is similar to designing optimal neural network structure. In the following subsections, we discuss important concepts which inform decision-making while designing an ideal multi-agent LLM system for the task at hand.

2.3.1 Tool use

Tool use has been a critical part of the development and evolution of humankind and is considered a unique characteristic of human beings as intelligent species[24]. For humans, tools are objects with specific functions and we mostly learn tool use through

²OpenAI o1 model can 'think' through problems before it responds, much like a person would: <https://openai.com/index/learning-to-reason-with-llms>

observation and demonstration. More complex tools like computers or programming languages usually come with a manual or tutorial that offers comprehensive information on their features and how to use them effectively. Similarly LLMs have been shown to comprehend how to use tools provided via a text interface through in-context learning[25]. This can be achieved by either zero-shot prompting where the language model is provided names of available tools, their descriptions, possible parameters, input and output formats, or by few-shot prompting where the model is provided concrete tool use demonstrations as examples. The language model takes in a set of available tools, few-shot demonstrations, and a task definition and outputs the name of the tool alongside the input parameters in JSON format. The LLM output is then processed to execute the tool suggested by the language model. Most recent LLMs are additionally trained on tool use instructions to further improve their tool use capability[26, 27, 28].

2.3.2 Chain of Thought

In-context learning has been shown to perform well on a wide array of NLP tasks[11] but it fails for tasks that require multiple steps and more complex reasoning[29]. To overcome this problem a technique called Chain of Thought has been introduced[21]. During Chain of Thought prompting the language model is provided intermediate reasoning traces inside each few-shot example. This causes LLMs to generate their "thinking process" before generating the final answer for the task. Chain of Thought has been shown to substantially improve performance of LLMs across various tasks, such as arithmetic reasoning, commonsense reasoning, and symbolic reasoning compared to simpler few-shot prompting[21].

2.3.3 ReAct

The ReAct study[25] introduces a novel approach to LLM agent reasoning by interleaving reasoning traces and task-specific actions. This approach allows the model to dynamically generate verbal reasoning traces that help guide its selection and execution of tools, while simultaneously using outputs from those tool interactions to refine and update its reasoning. ReAct enables large language model agents to solve complex tasks more effectively, especially in scenarios that require both strategic planning and interaction with external resources. Figure 3 demonstrates how ReAct helps to successfully complete a complex question-answering task compared to Chain of Thought and tool use baselines.

2.4 Retrieval Augmented Generation

LLMs, despite their impressive capabilities, can suffer from limitations related to their internal knowledge due to static training that can lead to outdated, incomplete, or poten-

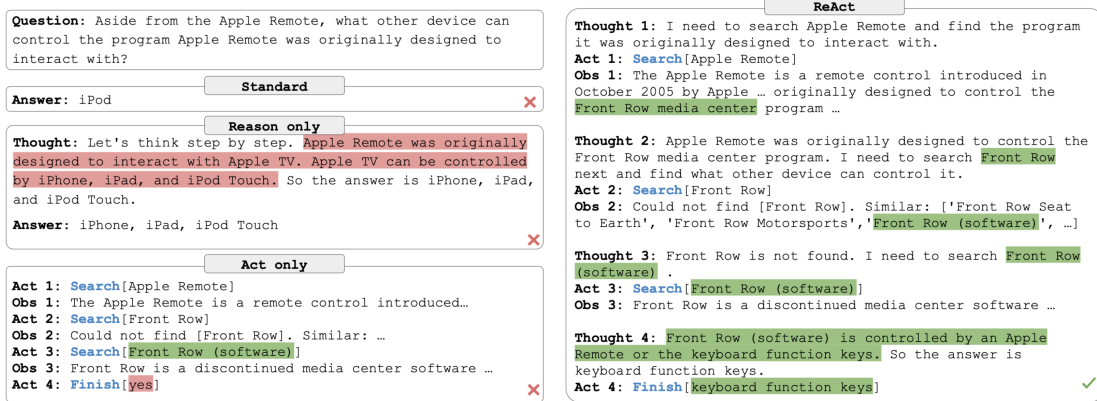


Figure 3. Complex question answering task example solved by ReAct. The Chain of Thought baseline suffers from hallucination (in red) as it is not grounded to external environments to obtain the required knowledge, and has to rely on limited internal knowledge. The tool use only baseline(Act) suffers from a lack of reasoning, unable to produce the final answer despite having the same actions and observations as ReAct. ReAct solves the task by giving a factually correct answer and providing interpretable reasoning traces. [25]

tially inaccurate information. These models essentially contain snapshots of knowledge frozen at the time of their training, which means they struggle with recent events, lack the ability to update their internal knowledge, and can sometimes generate confident-sounding but incorrect responses based on their training data. This problem is more evident when LLMs are applied to domains where the required knowledge is not publicly available and, therefore not in their training data. Retrieval Augmented Generation or RAG[30] effectively addresses these challenges by dynamically integrating real-time, relevant external information into the model’s context. In a RAG-powered AI system, when a query is received, the system first retrieves the most relevant documents or facts from an external knowledge base and then uses these retrieved documents to ground and inform the response generation of the LLM. This approach allows the language model to access up-to-date, domain-specific information that was not part of its original training, significantly improving the accuracy, relevance, and reliability of its outputs. Figure 4 illustrates a simple RAG-powered system.

3 Methodology

We introduce AgenticNER, a domain-agnostic NER framework that is implemented as a collaborative multi-agent system. In the following subsections, we discuss the

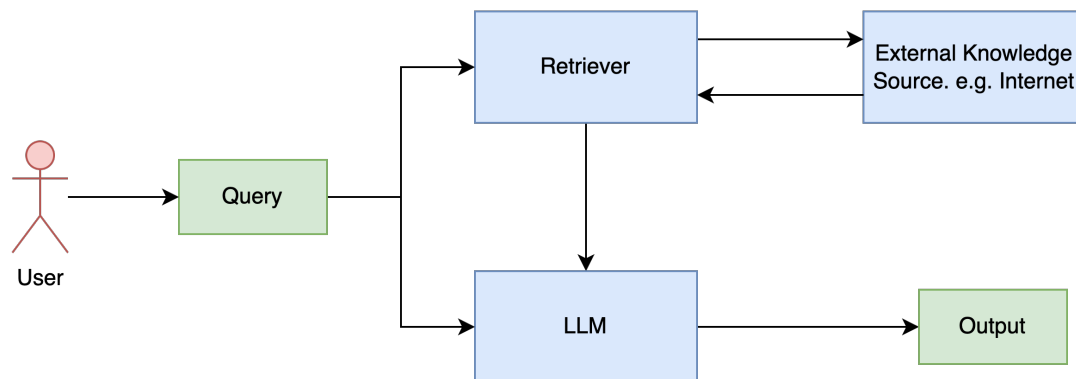


Figure 4. A simple RAG powered system. The user query is sent to a retriever to fetch relevant information from an external source. Information retrieved by the retriever is sent to LLM alongside the user query to get knowledge-grounded output.

architecture of AgenticNER and evaluation of this framework on benchmarks in 4 different domains.

3.1 AgenticNER

NER systems solely utilizing few-shot learning capability of LLMs excel at generic NER benchmarks but they struggle in domain-specific scenarios since LLMs’ internal knowledge is limited to the dataset they are trained on. They lack nuanced understanding of a human expert in use cases where there are granular entity types and the subtle differences between entity types in a particular domain can not be captured well by limited internal knowledge of general-purpose LLMs. LLM based NER systems suffer from hallucinations, confident-sounding but wrong outputs, that can not be trusted where precision is crucial and cost of false positive answers is big. NER models which are trained or fine-tuned for domain-specific purposes can mitigate these problems but such systems require a substantial amount of labeled data to build. AgenticNER attempts to overcome these limitations and trade-offs by utilizing a multi-agent design on top of in-context learning.

AgenticNER consists of 4 high-level components: a team of agents, a metaprompter, a chat supervisor and a grounding engine. The team of agents is composed of a tagger, a reviewer, and a researcher agent working collaboratively towards correctly tagging the input sequence. For agentic collaboration, we implement group chat communication protocol[31] with slight modification. In this communication pattern, each agent can write to and read from a common message pool, analogous to how people would communicate in a group chat when using an instant messaging application. The chat supervisor manages the group chat by selecting the next agent to speak based on the status of the

conversation and progress on the task. In the group chat, each agent is assigned a specific role and acts according to the prompts created by the metaprompter. The user provides the entity configuration, domain definition, and examples of valid inputs and expected outputs, which the metaprompter uses to generate a prompt for the tagger agent and deterministically construct a prompt for the reviewer agent. The research agent receives a static prompt. The research agent is also equipped with a search tool to query the internet and gather information necessary for completing the task. The entity configuration provided by the user contains names of entity types and their brief descriptions. After the agents reach a consensus on the final response, the grounding engine verifies or rejects the collaborative output of the agent team. The grounding engine uses a knowledge base provided by the user, which contains a set of phrases associated with potential entity types. Figure 5 demonstrates high-level components of AgenticNER and their interaction with each other.

To implement the group chat communication pattern we use AutoGen³, an open-source framework that simplifies the creation of event-driven agentic AI systems. As a search tool utilized by the research agent, we use Tavily Search API⁴, a real-time search engine designed specifically for large language models.

Below, we dive deep into each component of AgenticNER.

3.1.1 Metaprompter

Meta-prompting is a technique where a large language model is used to create precise and focused prompts for other LLMs. The Metaprompter component operates by taking examples of input-output pairs and an entity configuration as input from the user which includes entity types, their brief descriptions, and domain definition. While the component is named Metaprompter, it handles prompt creation through two different approaches, employing LLM-based meta-prompting for the tagger agent and deterministic template-based construction for the reviewer agent. To enable prompt generation across various domains for the tagger agent, we established a foundation by manually creating a base prompt. This base prompt was developed through an iterative refinement process using the development set from the Genia[32] benchmark, where we evaluated and improved the prompt’s effectiveness for few-shot NER based on evaluation results on the development set. The base prompt, developed for entity types in the molecular biology domain, serves as a learning example for the Metaprompter. By seeing how this prompt is structured for the Genia entity types, the Metaprompter can generate similar well-structured prompts for the tagger agent when given new domains and entity configurations. The Metaprompter prompt which contains the base prompt developed for

³<https://github.com/microsoft/autogen>

⁴<https://tavily.com/>

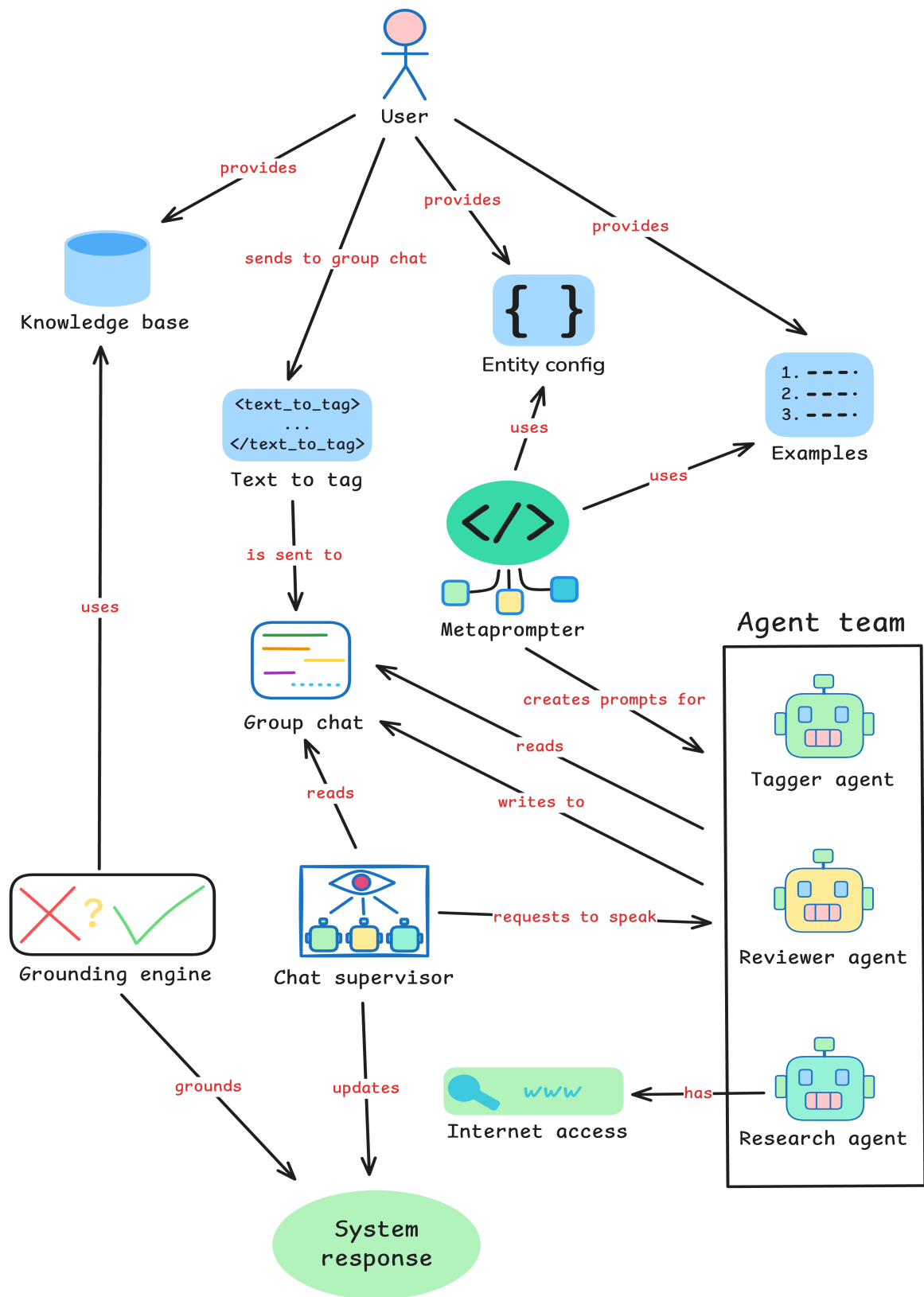


Figure 5. Components of AgenticNER and their interaction

Genia benchmark as an example can be found at github.com⁵.

The prompt for the reviewer agent is constructed based on a template. The template takes in the domain definition, entity descriptions, and example outputs. The body of the prompt template uses Chain of Thought and ReAct methods instructing the reviewer agent to think step by step critically about the output of the tagger agent and engage the research agent when it is not sure about an entity, concept, or fact. The prompt template for the reviewer agent can be found at github.com⁶.

3.1.2 The agent team

The agent team works collaboratively to accurately tag all named entity mentions in the input text provided by the user. The agents communicate using XML style tags, both to express their intents and to mark entities with their correct types. For example, in the sentence "Functionally, <protein>galectin-3</protein> was shown to activate <protein>interleukin-2</protein> production in <cell_type>Jurkat T cells</cell_type>.", different entity types are marked with their corresponding XML tags. The primary responsibility of the tagger agent is to identify all entities in the given text and present the tagged text within <output> tags. When needed, the tagger can request assistance from the research agent by asking questions within <search> tags. The reviewer agent analyzes the output of the tagger and provides feedback using <feedback> tags, or signals approval by outputting "APPROVED!" if the output of the tagger agent is satisfactory. Like the tagger, the reviewer agent can also seek help from the research agent using <search> tags. If the tagger agent disagrees with the reviewer's feedback, it can contest feedback by presenting its objection within <objection> tags, initiating a debate with the reviewer agent.

The research agent serves as an information source for both the tagger and reviewer agents and it uses its access to a search tool, essentially the whole internet, to answer their questions. Similar to how a human researcher would approach a question, the research agent can perform multiple searches using its search tool to thoroughly investigate and answer a single question. The prompt for the research agent can be found at github.com⁷. While all three agents participate in the same group chat, a crucial architectural decision was made to limit the researcher agent's visibility to only the messages specifically addressed to it through <search> tags. This limitation was implemented after initial experiments revealed that full conversation access caused the researcher agent to deviate from its intended role, attempting to provide feedback on the tagger's output or address

⁵<https://github.com/narasul/agent-ner/blob/e388b5e2104416f795874b9906aac9df30782840/src/ner/prompts.py#L248>

⁶<https://github.com/narasul/agent-ner/blob/e388b5e2104416f795874b9906aac9df30782840/src/ner/prompts.py#L403>

⁷<https://github.com/narasul/agent-ner/blob/e388b5e2104416f795874b9906aac9df30782840/src/ner/prompts.py#L509>

the reviewer's feedback instead of focusing solely on its research responsibilities. This finding aligns with previous work which reports that multi-agent debates using sparse communication topology can achieve superior performance compared to systems where each agent can bidirectionally communicate with all other agents[33]. Both tagger and reviewer agents, however, can see all of the message history in the group chat, including messages from the research agent.

3.1.3 Chat supervisor

The Chat supervisor manages turn-taking in the group chat by analyzing the content and source of each message to determine which agent should speak next. Upon receiving a message, the supervisor first checks if it's from the user. If it is, the supervisor requests the tagger agent to speak, initiating the entity tagging process. For messages from agents, the supervisor follows a decision tree based on the message content and the speaking agent. When processing a message from the tagger agent, the supervisor first checks for the presence of <output> tags. If <output> tags are present, the system response is updated with the latest tagged text, and the reviewer agent is requested to speak. If instead, the message contains <objection> tags, the supervisor requests the reviewer agent to speak to address the objection. For messages containing <search> tags, the research agent is requested to speak to provide information. If none of these tags are present, the tagger agent is requested to speak again.

When handling reviewer agent messages, the supervisor follows a similar pattern: <search> tags trigger the research agent to speak, <feedback> tags prompt the tagger agent to respond, and if "APPROVED!" is present, the system output is sent to the Grounding Engine. If none of these conditions are met, the reviewer agent is asked to speak again. For researcher agent messages, the supervisor simply requests the previous speaker who asked for help from the researcher agent to continue the conversation. If the message is from the Grounding Engine, which is a deterministic system component responsible for validating the final output, the supervisor checks whether the output was approved. When the Grounding Engine approves the output, the conversation ends. However, if the output is not approved, the supervisor requests the tagger agent to speak again to address the feedback from the Grounding Engine. To prevent endless agent interactions, the supervisor also enforces a maximum limit of 10 total speaking turns across all agents, after which the conversation is terminated and the last output from the tagger agent becomes the system's final response. The flow diagram in Figure 6 describes the decision-making process of Chat supervisor.

An example conversation in the AgenticNER group chat can be found in Appendix II.

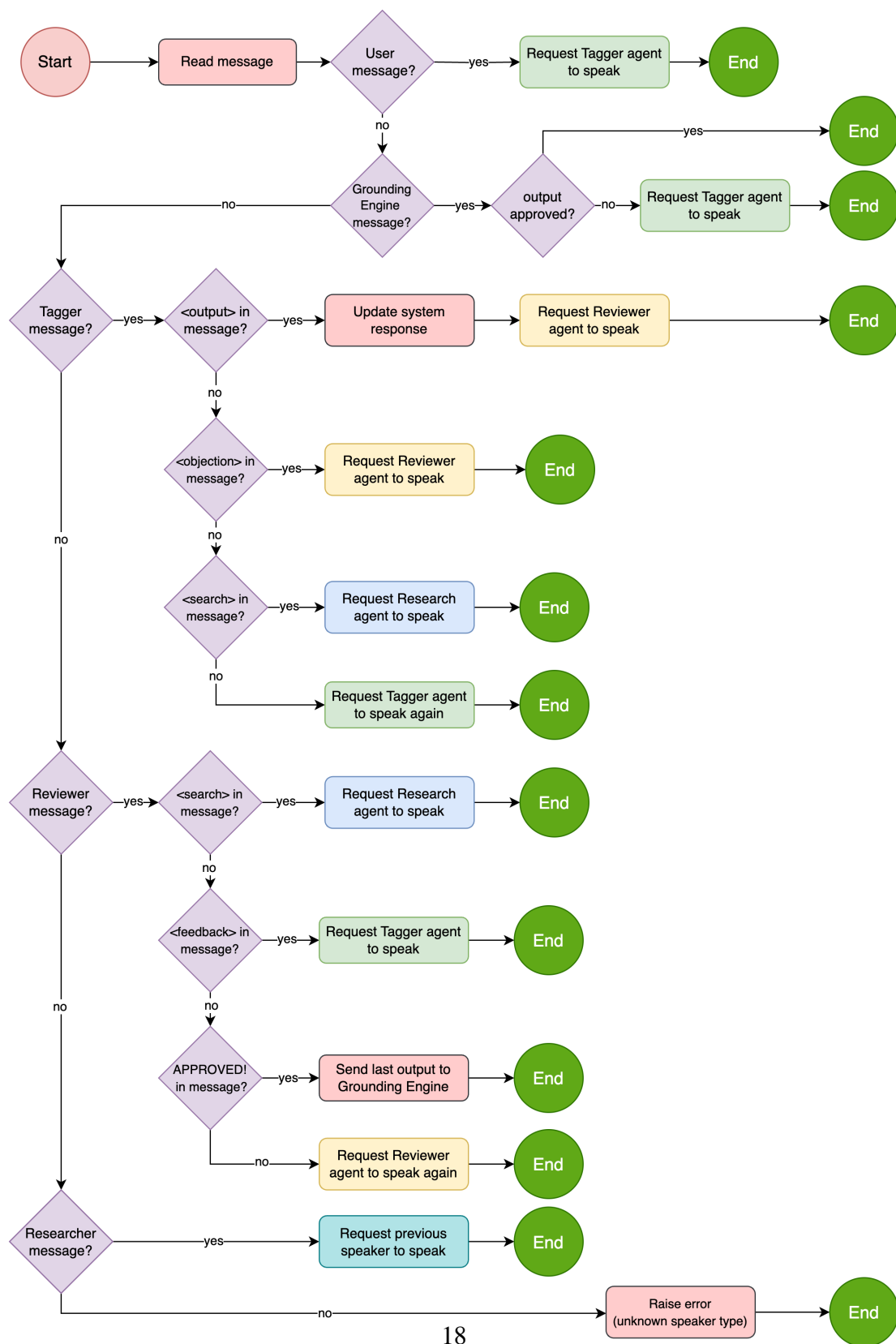


Figure 6. Flow diagram describes how Chat Supervisor selects the next group chat participant to speak

3.1.4 Grounding engine

The Grounding Engine serves as a verification mechanism to address the inherent limitations of LLMs, particularly their tendency to hallucinate due to limited domain-specific knowledge. Our implementation employs a user-provided knowledge base and allow the tagger agent an opportunity to revise its output based on feedback from the grounding engine. The knowledge base contains phrases and words associated with one or more entity types, establishing likely entity-type associations. The engine operates by verifying the tagger agent’s final output against this knowledge base. The output receives approval in two scenarios: either when the tagged entities are not present in the knowledge base, or when the assigned entity types align with those specified in the knowledge base. In cases where the tagger assigns an entity type that conflicts with the associations in the knowledge base, the Grounding Engine generates natural language feedback using a predefined template. This feedback is then provided to the tagger agent for consideration. Importantly, the grounding process occurs only once - if the tagger agent chooses to maintain its original output based on contextual understanding despite receiving feedback, the output is not subjected to further grounding attempts.

This soft grounding approach acknowledges the context-dependent nature of NER while still providing a mechanism to incorporate external knowledge for improved precision. While similar to VerifiNER[34], our grounding approach differs in that we do not use LLMs to create knowledge-grounded evidence from the knowledge base nor do we use consistency voting to determine the final output after grounding. Instead, we use the evidence from the knowledge base directly to create feedback for the tagger agent and allow only a single revision cycle from the tagger agent to determine the final system output. Therefore, our approach is more lightweight and suitable for multi-agent systems where latency and costs are already high. Although we do not exclude the possibility of incorporating VerifiNER’s approach in future iterations of our framework, the additional cost and latency overhead of multiple LLM calls for consistency voting would need to be carefully evaluated against potential performance benefits.

3.2 Evaluation

To evaluate the effectiveness of AgenticNER we conducted experiments across several domains and benchmarks. In this subsection, we present our evaluation methodology.

3.2.1 Evaluation metrics

NER, as a sequence labeling task, requires evaluation against labeled ground truth sequences. The standard evaluation metrics for such tasks are Precision, Recall, and F1-score, which are calculated using true positives, false positives, and false negatives. In the context of NER, a true positive occurs when the system correctly identifies both

the token and type of an entity in the sequence, a false positive occurs when the system incorrectly identifies non-entity token as an entity or assigns the wrong type to entity token, and a false negative occurs when the system fails to identify an entity that exists in the ground truth. These metrics can be calculated using three different averaging methods: micro, macro, and weighted. Macro averaging calculates these metrics independently for each entity type and takes their unweighted mean, which means poor performance on minority classes significantly impacts the final score. Weighted averaging follows a similar approach but weighs contribution of each entity type by its support - number of instances, effectively giving more importance to majority entity types. Micro averaging, on the other hand, aggregates all true positives, false positives, and false negatives across all entity types before calculating the metrics. This approach is well-suited for imbalanced multi-class classification problems like NER, as it provides a global assessment of system performance. For our evaluation, we utilize micro-averaged F1-score, Precision, and Recall, as they provide the most appropriate measure of overall system performance in the context of imbalanced entity distributions typically found in NER benchmarks. Micro averaged Precision P , Recall R and F1-score $F1$ can be calculated for N entity types using following formulas where TP , FP , and FN correspond to true positives, false positives, and false negatives respectively:

$$P = \frac{\sum_{i=1}^N TP_i}{\sum_{i=1}^N (TP_i + FP_i)} \quad (1)$$

$$R = \frac{\sum_{i=1}^N TP_i}{\sum_{i=1}^N (TP_i + FN_i)} \quad (2)$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (3)$$

These evaluation metrics each provide distinct insights into performance of a NER system. Precision measures what proportion of our predicted entities are correct, making it more important in applications where false positives are costly. For instance, in knowledge graph construction using NER, high precision is crucial since all false positives would propagate into the created knowledge graph, dramatically reducing its reliability. Recall, on the other hand, measures what proportion of actual entities in the text are successfully identified, making it critical in applications where missing an entity could have serious consequences, such as in security applications where failing to identify a threat mention could be critical. There is often a trade-off between these metrics - achieving high precision might come at the cost of lower recall and vice versa. This trade-off

occurs because being more conservative in entity prediction to achieve high precision typically means missing some valid entities, while being more liberal in predictions to achieve high recall often leads to more false positives. The F1-score, being the harmonic mean of precision and recall, provides a balanced measure of performance that takes into account both metrics. The choice of which metric to prioritize often depends on the specific application of NER, but the F1-score provides a solid general-purpose metric for comparing different NER systems.

3.2.2 Challenges with evaluation

Conventional NER systems approaching NER as sequence labeling task can produce outputs in different formats, CoNLL, IOB, and IOB2 being more common. For convenience, we have converted the ground truth labels across all four benchmarks into IOB2 format. In IOB2 format, each token in the output sequence is either prefixed with 'B-', indicating the beginning of an entity, prefixed with 'I-', indicating the token is inside an entity, or is just the token 'O', indicating the token is outside any entity.

As a system based on generative LLMs AgenticNER approaches the NER task differently. Instead of assigning labels to each token from the input sequence, it instructs the tagger agent to generate an output where entities are marked with XML-style tags inside original input sequence. For instance, given the input sequence *"Finally, no transcription of the RAG-1 gene could be detected in all NK cell lines or clones analyzed."*, AgenticNER generates *"Finally, no transcription of the <DNA>RAG-1 gene</DNA> could be detected in all <cell_line>NK cell lines</cell_line> or clones analyzed."* while the ground truth for this sequence in IOB2 format would be ['O', 'O', 'O', 'O', 'O', 'O', 'B-DNA', 'I-DNA', 'O', 'O', 'O', 'O', 'O', 'B-cell_line', 'I-cell_line', 'I-cell_line', 'O', 'O', 'O', 'O'].

This divergence between generative output and traditional sequence labeling ground truth found in NER benchmarks creates an interesting evaluation challenge. The most intuitive approach to bridge this gap would be to tokenize the generated output and convert the tokenized output to IOB2 format looking at the boundaries defined by the XML tags. However, while testing this approach we discovered a peculiar behavior of LLMs - they do not always faithfully reproduce the exact input sequence when adding tags. LLMs often make subtle modifications to the input text, such as correcting grammar errors or adjusting punctuation. This characteristic of LLMs causes the output sequence to differ in length from the input sequence and consequently ground truth rendering comparison between the output of AgenticNER and ground truth invalid during evaluation. To investigate the extent of the problem we conducted an experiment by running AgenticNER on the BUSTER evaluation set and measuring the Levenshtein distance[35] between each input sequence and output sequence(after stripping all XML tags). Interestingly, we found a linear correlation between length of the input sequence(in characters) and the Levenshtein distance between input and output sequences suggesting

that longer input sequences result in more modifications from the large language model. Figure 7 demonstrates this correlation with scatter plot and linear regression line.

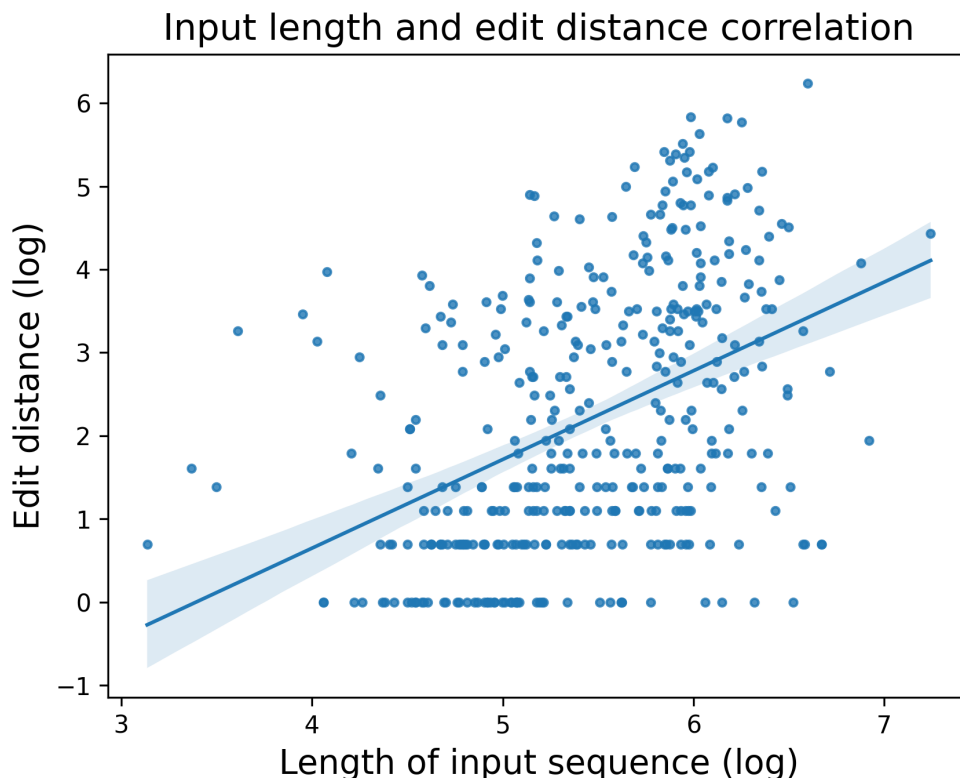


Figure 7. The correlation between length of the input sequence and the Levenshtein distance between input and output sequences(Logarithmic scale) can be clearly seen in the scatter plot.

During initial experimentation on AgenticNER we explored various alternatives to XML tagging, such as asking the LLM to output JSON with entity spans as keys and entity types as values, or even requesting the LLM to directly output boundary indexes for entities. However, we found out that LLMs are more reliable at handling NER task by generating XML tags compared the other formats we have considered. Therefore, the fundamental challenge remained in evaluating the system that generates modified versions of the input text which must stay equal in length to the ground truth it is being evaluated against.

To address this challenge, we developed a heuristic approach. We extract entity spans(words or phrases corresponding to an entity) from XML tags in the LLM output, tokenize these spans, and then identify indices of the longest matching contiguous

subsequence between the extracted span tokens and original input tokens, with the constraint that the subsequence should start from the first token of the span. While this solution works effectively in most cases, it makes a naive assumption that no two identical token sequences in the same sentence can have different entity types. For example, if we are required to tag entities in the sentence "*Silicon Valley, a comedic take on tech culture in Silicon Valley*", AgenticNER can output "`<work_of_art>Silicon Valley</work_of_art> is a comedic take on tech culture in <location>Silicon Valley</location>`", but our heuristic approach would fail to correctly convert the output to IOB2 format.

We acknowledge the limitation of this heuristic approach and use it as a practical solution to a novel challenge of evaluating generative NER systems. To our surprise, this challenge has not received any attention in existing literature we have reviewed. Growing adoption of LLMs and in-context learning for NER tasks suggests that developing more robust evaluation methodologies for generative NER systems will be increasingly important. We have decided to keep this challenge outside the scope of this thesis and consider it a promising direction for future work.

3.2.3 Datasets

To validate the domain-agnostic capabilities of AgenticNER and evaluate its performance across varying levels of complexity, we selected benchmarks from four distinct domains: Genia[32] in biomedical domain, BUSTER[36] in financial domain, Astro-NER[37] in astronomy domain, and MusicRecoNER[38] in music and entertainment domain.

Genia Genia is a collection of biomedical paper abstracts annotated within the scope of the GENIA project[32]. It was created in 2003 to enable development and evaluation of text mining and information extraction systems in the domain of molecular biology. Genia includes a term annotation subcorpus that contains 2000 abstracts annotated with 5 named bio-entities: protein, DNA, RNA, cell type, and cell line. We are using a version of this subcorpus available in the HuggingFace repository⁸. In this version, the abstracts are preprocessed by splitting them into individual sentences, where each sentence becomes a separate entry in the dataset. Each entry consists of four components: the target sentence itself as a sequence of tokens, its preceding context, its following context, and entity labels for the target sentence. This format makes entity recognition easier for NER systems by shortening the output sequence while preserving the contextual information around each sentence. The test set of this dataset contains 1854 entries, of which we have only used a random sample of 500 entries to minimize costs associated with commercial large language model vendors.

⁸<https://huggingface.co/datasets/Rosenberg/genia/viewer/default/test>

BUSTER BUSTER or BUSIness Transaction Entity Recognition dataset, published in 2023, is a NER benchmark focusing on entities related to business transactions in financial documents. The dataset consists of two parts, a manually annotated gold corpus containing 3779 documents and a silver corpus with 6196 documents labeled by a NER system based on a RoBERTa model. The gold corpus is divided into 5 equal folds, of which we have used one fold for evaluation. The financial documents in the dataset are rather long with average sequence length of 896 tokens. Therefore we have preprocessed the documents using the same approach applied to the Genia dataset by splitting each document into individual sentences, where each sentence becomes a separate dataset entry accompanied by its surrounding context from the source document and its original entity labels. We filtered out all entries that had no entities present and randomly sampled 500 entries for our final evaluation set. There are six entity types in BUSTER: buying company, selling company, acquired company, legal consulting company, generic consulting company, and annual revenues.

MusicRecoNER MusicRecoNER is a NER benchmark created in 2023 focusing on music recommendation queries collected from Reddit. The dataset consists of informal, noisy single-sentence queries in English that were collected from human-human interactions designed to simulate conversations between users and music recommendation assistants. Each query is annotated with two types of entities: Artist and Work of Art. For evaluation, we have used all 601 entries of the original test dataset as already smaller size of this dataset eliminated the need for further sampling. We also have not performed any preprocessing on the original sentences.

Astro-NER Astro-NER is a corpus of 5000 annotated astronomy article titles created in 2024 to address the challenge of developing NER models for specialized scientific domains. The titles are mainly from the astrophysics subfield, with a particular emphasis on astroparticle physics. The dataset contains 12 distinct entity types - AstrObject, AstroPortion, ChemicalSpecies, Instrument, Measurement, Method, Morphology, PhysicalQuantity, Process, Project, ResearchProblem, and SpectralRegime - with ResearchProblem and Method being the most frequent with 3801 and 3169 mentions respectively. We have split the original dataset into test and development sets and similarly to Genia and BUSTER datasets we have randomly sampled 500 entries from the test set for evaluation.

Table 1 shows distribution of entity types in each evaluation set we are using.

3.3 Baseline

As a baseline we implemented a few-shot single LLM call method that generates entity-tagged output sequence using a single invocation of large language model. This approach is equivalent to using the tagger agent in isolation, without the ability to receive feedback

Genia	
Entity type	Support
Protein	709
DNA	339
RNA	25
Cell Type	176
Cell Line	125
BUSTER	
Buying Company	425
Selling Company	110
Acquired Company	322
Legal Consulting Company	28
Generic Consulting Company	68
Annual Revenues	48
MusicRecoNER	
Artist	303
Work of Art	208
Astro-NER	
Astronomical Object	14
Astronomical Portion	15
Chemical Species	93
Instrument	79
Measurement	46
Method	332
Morphology	32
Physical Quantity	48
Process	141
Project	14
Research Problem	361
Spectral Regime	17

Table 1. Distribution of entity types in the evaluation sets

from the reviewer agent, assistance from the research agent, or feedback from the grounding engine. To enable fair comparison with AgenticNER, the baseline uses the same tagger agent prompt created by the Metaprompter component. Naturally, this prompt includes references to the reviewer and researcher agents. We considered creating a separate Metaprompter base prompt for the baseline that would remove these agent references, however, several factors influenced our decision to maintain the same prompt. First, using a modified base prompt for Metaprompter could introduce unwanted variance in the generated prompts, making it difficult to attribute performance differences specifically to the absence or presence of the multi-agent architecture. Second, the few-shot examples provided to both systems contain only input-output pairs and are sampled using the same methodology to make sure the task demonstrations are identical regardless of the presence of agentic architecture. And finally, our evaluation runs showed that in single LLM invocation mode, the tagger agent never attempts to utilize the research agent’s capabilities, and without any feedback from the reviewer agent it does not acknowledge the existence of the reviewer. As a result, we decided to use few-shot single LLM call method with the same tagger agent prompt as baseline to evaluate the effectiveness of AgenticNER’s collaborative multi-agent design.

3.4 Experiment setup

In this section, we discuss the experiment setup choices we have made for evaluation of AgenticNER such as the choice of LLMs, example sampling strategies, model parameters, and the source of user-provided external knowledge bases.

3.4.1 Large Language Models

For evaluations, we are using Claude 3.5 Sonnet and Claude 3.5 Haiku from Anthropic Claude model family⁹.

At the time of writing, Claude 3.5 Sonnet is state-of-the-art large language model that sets new industry benchmarks for several complex reasoning tasks including graduate-level reasoning(GPQA[39]), undergraduate-level knowledge(MMLU[40]), and coding proficiency(HumanEval[41]). It shows significant improvement over similar models in understanding nuance and complex instructions. When instructed and provided with the necessary tools Sonnet 3.5 can independently write, execute, troubleshoot, and edit code demonstrating complex reasoning capabilities in agentic setting.

Claude 3.5 Haiku is significantly smaller and, therefore faster and cheaper model than Claude 3.5 Sonnet using the same underlying architecture. It improves over the largest model of the previous generation of Claude models, Claude 3 Opus, on many intelligence benchmarks. Claude 3.5 Haiku provides improved instruction following and accurate

⁹<https://www.anthropic.com/news/3-5-models-and-computer-use>

tool use capabilities with low latency, making it ideal for use in a multi-agent system. At the time of writing, Sonnet costs 3.75 times more than Haiku on the Anthropic API.

The knowledge cutoff date for Claude 3.5 Sonnet is April 2024, meaning its training data includes publicly available information up until that time. For Claude 3.5 Haiku, the knowledge cutoff date extends to July 2024[42].

In all of our evaluation runs, we use Claude 3.5 Sonnet as Metaprompter to generate prompts for the tagger agent. We are running evaluations using both Claude Sonnet 3.5 and Claude Haiku 3.5 as the language model backing the individual agents. By analyzing a state-of-the-art large language model alongside a more efficient but less capable model, we aim to understand how the reasoning capabilities of the underlying language model impact the overall performance and behaviors of agents in a collaborative multi-agent setting.

3.4.2 Example sampling

Few-shot learning works by providing examples of task demonstrations to a large language model during invocation. Several sampling strategies exist that aim to provide the most relevant in-context examples to the large language model to improve performance, with the most naive strategy being sampling N number of input-output pairs from available labeled data. More sophisticated methods usually work by querying a database of input-output pairs such that the sampled examples are semantically or lexically close to the task input. Wang et al.[12] reports comparable performance to fully supervised baselines in NER using in-context learning and selecting the most similar sentences from the training set using kNN search over entity-level embeddings.

We position AgenticNER as a domain-agnostic framework and assume a lack of labeled data in the domain the user is interested in. We contend that if such a dataset existed the user would be able to build a lightweight supervised model that can perform comparably or better, is not autoregressive in nature, and consequently has orders of magnitude less inference cost. We do not compare AgenticNER with supervised baselines, hence, we use a random sampling method in evaluation of the framework. More specifically, we randomly sample a fixed number of input-output pairs from the available training data in each benchmark so that each entity type is mentioned at least once. This is similar to the method known as N-Way K-Shot sampling in literature, where N is the number of entity types, and K is the number of examples for each entity type.

For Genia, BUSTER, and MusicRecoNER benchmarks, we sample a total of 3 examples with the constraint that each entity type in the benchmark is covered in those 3 examples. For Astro-NER, we sample a total of 12 examples with the same constraint. This is because Astro-NER has 12 different entity types, and is generally considered a harder benchmark.

3.4.3 Knowledge Base

The grounding engine in AgenticNER uses a knowledge base provided by the user, which contains a set of phrases associated with likely entity types. For evaluation, we construct the knowledge base using the training set of each benchmark. On the surface, this methodology conflicts with the assumptions made while choosing the example sampling strategy. However, entity spans associated with likely entity types are not equivalent to input-output sequence pairs in labeled training data. Such data is often naturally occurring and can be obtained with relative ease compared to a training dataset which requires significant manual effort to create. For instance, The Unified Medical Language System¹⁰ contains a large database that includes over 1 million biomedical concepts and terms with their associated semantic type which can be used as a knowledge base for biomedical NER.

3.4.4 Temperature and non-determinism

Temperature is an inference hyper-parameter that controls the randomness or determinism in a large language model’s token generation process. During inference, LLMs compute a probability distribution over their vocabulary for every next token prediction step. Mathematically, temperature T is applied by dividing the logits, raw prediction vector from the model, by T before applying the softmax operation which converts the prediction vector into a probability distribution. When T is smaller than one, the distribution becomes more peaky, causing the model to favor high-probability tokens more strongly. As T approaches 0, the model consistently selects the most probable token. When T is greater than 1, the distribution becomes more uniform, increasing the likelihood of selecting lower-probability tokens. In practice, higher temperature results in more random LLM output, while lower temperature results in more deterministic outputs.

Theoretically, setting the temperature to 0 means fully deterministic outputs, therefore we have done so in all of our evaluation runs. However, due to non-determinism attributed to floating point arithmetic in GPUs[43] LLMs are not guaranteed to always produce the same output given the same input.

Deterministic outputs are important for reliability and reproducibility of any empirical study. One way to introduce stability into evaluation of a non-deterministic system would be running the evaluation multiple times and reporting the average score across all evaluation runs. However, such method is cost-prohibitive in our case due to high number of LLM calls associated with a multi-agent system.

To understand the impact of non-determinism on our task and choice of LLMs we sampled 50 entries from Genia test set and repeated the evaluation 25 times. The resulting standard deviation for precision, recall and F1 score across 25 evaluation runs was **0.01**,

¹⁰<https://www.nlm.nih.gov/research/umls/index.html>

0.007, and **0.008** respectively. As a result, we decided to report evaluation results based on a single evaluation run for each benchmark.

4 Results

Table 2 shows evaluation results across all 4 benchmarks. On a higher level, we observe that AgenticNER consistently performs better than single LLM call baseline in 3 out of 4 benchmarks in terms of F1 score. This holds true when using both Claude Sonnet 3.5 and the less capable Haiku model. In one of the benchmarks, Astro-NER, AgenticNER performs worse than the baseline across all 3 measures, precision, recall, and F1 score.

When using Claude 3.5 Haiku, a smaller and faster LLM, in Genia, BUSTER, and MusicRecoNER benchmarks AgenticNER improves over the baseline for precision and F1 measures, increasing precision significantly. However, when using Haiku, recall is always better with the single LLM call baseline across all 4 benchmarks.

Similar but slightly different trend can be seen when using the more capable Claude 3.5 Sonnet model. The difference is that AgenticNER performs better in terms of recall as well in Genia and MusicRecoNER benchmarks. Furthermore, in MusicRecoNER AgenticNER performs worse in terms of precision compared to when using Claude 3.5 Haiku model. We attribute the increase in recall in Genia and MusicRecoNER to the increased reasoning capabilities of Sonnet model. When backed with a more capable model the reviewer agent points out to entities in the input text that the tagger agent missed to tag. This behavior is hard to enforce through chain of thought prompting, which explains why the smaller model fails to increase recall through agentic interactions. The decrease in precision in MusicRecoNER benchmark can be explained by already high precision of single LLM call baseline. It appears that while the tagger agent initially identified most entities correctly, it subsequently changed its accurate responses based on invalid feedback from the reviewer

Astro-NER is a challenging benchmark with 12 different entity types. Here when using both Sonnet and Haiku models AgenticNER performs worse than the baseline. This limited evidence suggests that agentic architecture is not the best choice for every use case. Another observation we made is that the more capable models are more robust in agentic setting and are affected less by invalid feedback from the agent’s environment: AgenticNER reduces F1 score from 0.36 to 0.35(2.77% decrease) when using Sonnet model while using Haiku, the score drops from 0.35 to 0.32(8.57% decrease).

Finally, we draw the following conclusions from evaluation of Agentic NER in 4 different benchmarks:

- AgenticNER performs better in 3 out of 4 benchmarks proving multi-agent architecture to be promising approach to domain-agnostic NER.

Method	Precision	Recall	F1
Genia			
<i>Claude 3.5 Sonnet</i>			
Single LLM call baseline	0.59	0.51	0.55
AgenticNER	0.65	0.53	0.58
<i>Claude 3.5 Haiku</i>			
Single LLM call baseline	0.51	0.50	0.51
AgenticNER	0.58	0.48	0.53
BUSTER			
<i>Claude 3.5 Sonnet</i>			
Single LLM call baseline	0.60	0.75	0.67
AgenticNER	0.66	0.71	0.69
<i>Claude 3.5 Haiku</i>			
Single LLM call baseline	0.47	0.66	0.55
AgenticNER	0.50	0.64	0.56
MusicRecoNER			
<i>Claude 3.5 Sonnet</i>			
Single LLM call baseline	0.93	0.79	0.85
AgenticNER	0.90	0.83	0.86
<i>Claude 3.5 Haiku</i>			
Single LLM call baseline	0.83	0.86	0.84
AgenticNER	0.87	0.83	0.85
Astro-NER			
<i>Claude 3.5 Sonnet</i>			
Single LLM call baseline	0.37	0.35	0.36
AgenticNER	0.36	0.34	0.35
<i>Claude 3.5 Haiku</i>			
Single LLM call baseline	0.33	0.36	0.35
AgenticNER	0.33	0.31	0.32

Table 2. Evaluation results for AgenticNER in 4 different benchmarks

- Collaborative multi-agent architecture does not necessarily perform better than non-agentic baseline in all domains and use cases.
- More capable LLMs benefit more from multi-agent design while being affected less by adverse effects of invalid external feedback.

5 Ablation study

5.1 Variants

We compared 4 different variants of AgenticNER for the ablation study and analyzed the effect of each component on F1 score, precision, and recall measures separately, with the first variant being AgenticNER with all components. In the second variant, we simply removed the grounding engine and accept the final collaborative response of the multi-agent system without rule-based grounding. In the third variant, we removed internet access from the researcher agent and modified its prompt so that the agent responds to questions using its internal knowledge. We also modified the reviewer agent’s prompt by removing the part of the prompt where we tell that the researcher has internet access. In the final variant, we removed any mention of research agent from the prompts and removed the research agent completely from the system.

To minimize costs, we conducted all experiments using the smaller Claude 3.5 Haiku model.

5.2 Impact on Precision

Figure 8 shows the impact of each AgenticNER variant on precision gains (or losses) over the few-shot single LLM call baseline. In general, how different components of AgenticNER affect precision varies highly depending on the benchmark.

In the Genia benchmark, all variants demonstrate consistent positive impact on precision over the baseline. Removing the grounding engine appears not to have any impact compared to full AgenticNER variant, while removing internet access or the research agent equally improves precision marginally. This means that in Genia, the LLM’s internal knowledge suffices to complete the task. This is likely because Genia is an old corpus, created in 2003, and the LLMs are already trained on the data this corpus contains. Removing the research agent or research agent’s internet access improves the precision, which means in this benchmark extra information from the research agent hurts the system. This highlights a well-known shortcoming of LLMs: they cannot ignore information that they consider to be from an authoritative source. Several studies report that LLMs can be easily "distracted" by irrelevant context or misinformation[44, 45]. We have implemented AgenticNER in a way that should reduce such distractions - the tagger

agent is prompted to object to the feedback from the reviewer agent which it considers incorrect, and as a result, the effect of such distractions is minimal.

A similar trend to the Genia benchmark can be seen with the BUSTER benchmark. The grounding engine does not have any impact on precision. This is expected since the knowledge base for the grounding engine is built from the BUSTER training set, the entity types in BUSTER - company names and annual revenues - are high cardinality, and the entities are unlikely to be repeated across training and test sets since each entry in the dataset is an article about business transactions between companies. Similar to Genia, we also see negative impact of internet access and the researcher agent in BUSTER, with the difference that having internet access hurts precision more than having a researcher without internet access. We have observed that when the research agent has internet access, the reviewer agent often wants to learn about companies involved in the transaction from the researcher which is irrelevant for identifying companies involved in a transaction.

In MusicRecoNER, the pattern is different than in Genia and BUSTER. Here, the most optimal variant is the full AgenticNER system with all of the components. The precision drops significantly when the grounding engine is disabled, indicating lower cardinality of the entity types compared to BUSTER and Genia datasets. More interestingly, AgenticNER with the grounding engine removed performs worse than the baseline in terms of precision. This indicates that in this benchmark, the multi-agent system is more "distracted" by external information and feedback in the absence of rule-based

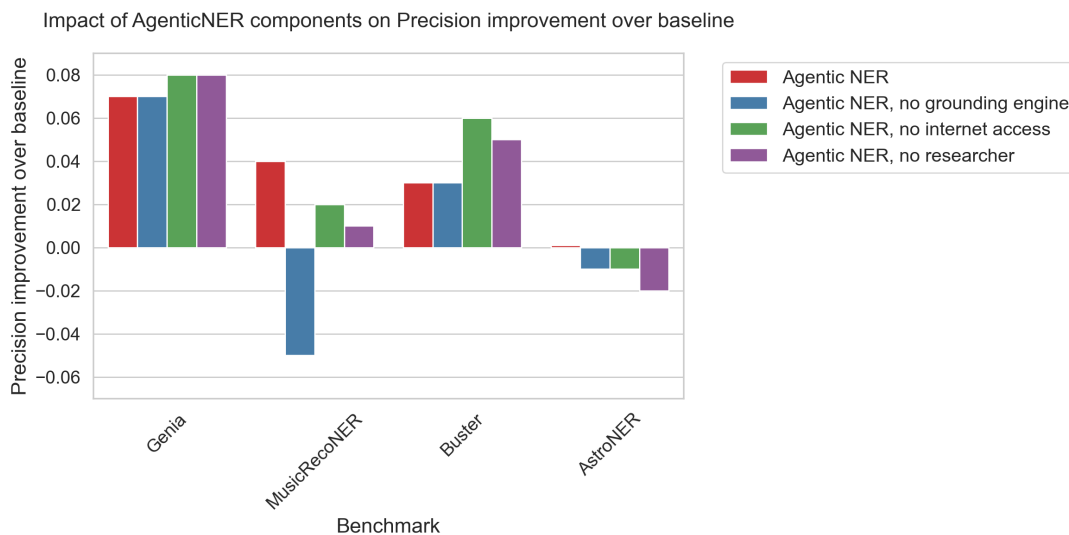


Figure 8. Impact of each AgenticNER component on precision improvement over few-shot single LLM call baseline.

grounding. When the grounding engine is present but the research agent is removed, the system still performs worse than the full AgenticNER system, which means the presence of the research agent is marginally helpful. Removing internet access without removing the research agent also performs worse than full AgenticNER, which indicates that the information from external sources was beneficial.

In Astro-NER, AgenticNER with all components does not impact precision compared to the baseline. However, removing the grounding engine, the internet, or the researcher agent decreases precision. The outcomes on Astro-NER match our expectations from AgenticNER components.

5.3 Impact on Recall

Figure 9 reveals interesting patterns in how AgenticNER components affect recall, with significant differences across benchmarks and deviations from impact on precision.

The grounding engine shows no impact on recall in all benchmarks but BUSTER, which aligns with its design purpose of filtering false positives rather than identifying new entities.

In the Genia benchmark, both full AgenticNER and the variant without the grounding engine show equal decrease in recall compared to the baseline. However, removing internet access improves recall back to baseline levels, and removing the research agent entirely leads to recall exceeding the baseline. This suggests that in an old benchmark in

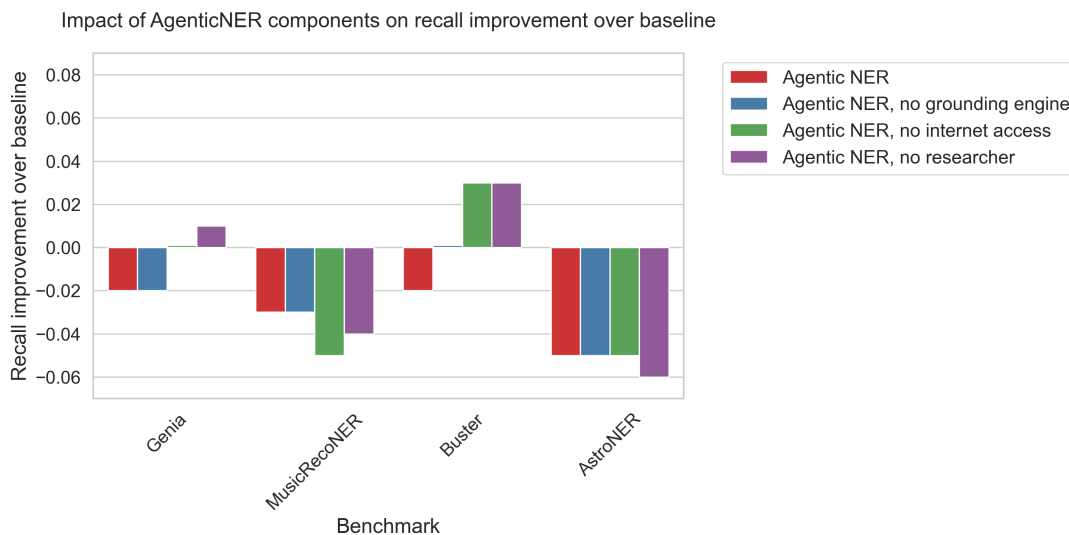


Figure 9. Impact of each AgenticNER component on recall improvement over few-shot single LLM call baseline.

any domain where all information is likely to be part of the LLM’s training data, external information, and feedback can diminish both precision and recall.

MusicRecoNER exhibits a similar pattern to Genia where full AgenticNER and the variant without grounding engine show equal recall performance below the baseline. However, unlike Genia, removing internet access results in the most significant recall decrease among all variants. Removing the researcher agent also decreases recall, but less severely than removing internet access alone. This indicates that in the music and entertainment domain, a research agent with internet access contributes more positively to recall than the one limited to internal knowledge.

BUSTER demonstrates a unique pattern distinct from other benchmarks. While full AgenticNER results in lower recall than baseline, removing the grounding engine improves recall to baseline levels. This improvement can be attributed to the grounding engine rejecting true positives when powered by an imperfect knowledge base. More interestingly, removing either internet access or the researcher agent leads to significant recall improvements above the baseline. This behavior is exemplified by a case we observed where the reviewer asked the research agent to confirm the revenue of a company being bought in a certain year. The research agent provided contradictory information to the revenue figure in the input text, causing the system to remove a correct annual revenue entity from the output.

In Astro-NER, all variants of AgenticNER show significant recall decreases compared to baseline, with the removal of the research agent causing the most severe drop. This uniform degradation in recall across all variants suggests that in domains with many specialized entity types, the complexity of multi-agent interactions and external knowledge integration can significantly reduce the system’s effectiveness.

5.4 Impact on F1 score

Figure 10 demonstrates how different variants of AgenticNER affect F1 score compared to the baseline across all benchmarks. Naturally, the results reflect the interaction between the impact of different components on precision and recall we discussed earlier.

In Genia, removing internet access or the research agent leads to the highest F1 score improvements among all variants, which aligns with our earlier observations about how external information can be detrimental to precision and recall in this older benchmark where LLM’s internal knowledge suffices to correctly tag entity types.

In MusicRecoNER AgenticNER with all components results in modest F1 score improvement over baseline, but removing the grounding engine causes a significant drop. This substantial decrease can be attributed to the previously observed precision drop in this variant, highlighting how beneficial rule-based grounding can be for overall performance in domains with lower cardinality entity types. Both variants without internet access and research agent perform marginally worse than the baseline but better than the variant without grounding engine.

BUSTER exhibits a unique pattern that reflects the interesting interplay between precision and recall in this benchmark. While full AgenticNER shows modest improvement over baseline, removing the grounding engine leads to slightly higher improvement. The most notable gains come from removing internet access or the researcher agent, which aligns with our earlier findings that external information tends to confuse the agents in this domain where identifying company names and annual revenues in text requires focusing on the input text rather than additional context.

In Astro-NER, the pattern is clear and consistent - all variants show F1 score decreases compared to baseline, with removal of the research agent causing the most severe drop. This uniform degradation in performance across all variants mirrors what we observed in both precision and recall analyses, reinforcing our earlier conclusion that in domains with many specialized entity types with subtle differences, the complexity of multi-agent interactions may be counterproductive.

Based on the findings we conclude that the effectiveness of different AgenticNER components on overall performance is highly domain-dependent, with some domains benefiting from full functionality while others perform better with limited external information and feedback. AgenticNER is a modular framework where users can choose to use a different knowledge base, turn off the grounding engine, or provide a different search tool to the researcher agent for access to an information source other than the internet based on their use case and domain. Therefore, the results of the ablation study do not undermine the utility of AgenticNER, but rather provide users with guidance on

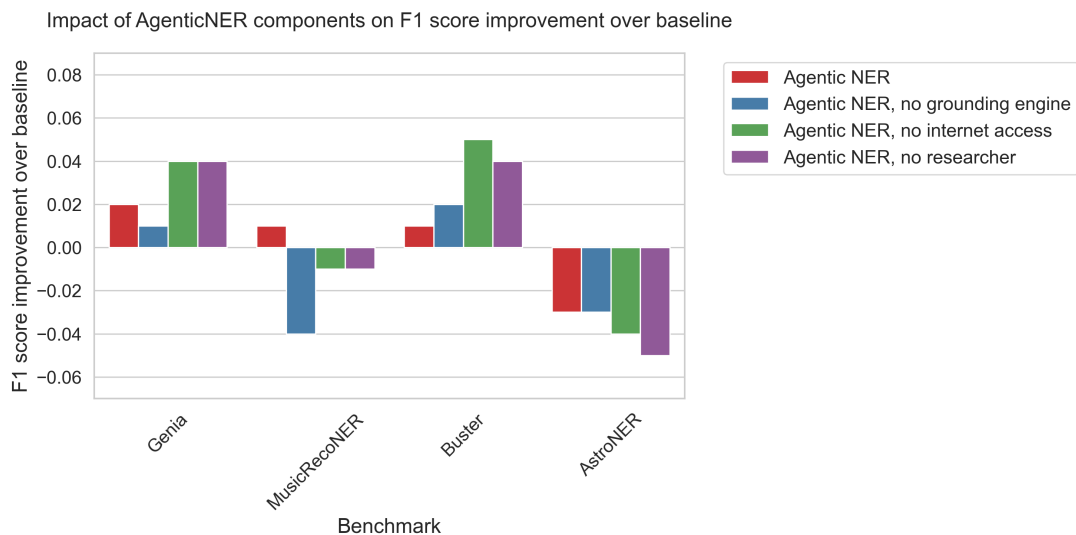


Figure 10. Impact of each AgenticNER component on F1 score improvement over few-shot single LLM call baseline.

which components might be most effective for their domain.

6 Limitations and future work

The evaluation of AgenticNER showed promising results across benchmarks in multiple domains; however, it also revealed the framework’s limitations and highlighted interesting areas for future research. In the following subsections, we discuss these limitations and potential future research directions, focusing on three key areas: improving recall performance, developing more robust evaluation methods for generative NER, and exploring better knowledge base integration techniques.

6.1 Improving recall

While AgenticNER demonstrates promising results in terms of precision, our evaluation reveals consistent challenges with recall performance across benchmarks. This limitation becomes particularly evident when using the more efficient Claude 3.5 Haiku model, where recall consistently lags behind the baseline. The multi-agent architecture, though effective at filtering false positives through multi-agent debate and grounding, may be overly conservative in entity identification. Future work should explore enhanced prompting strategies for the reviewer agent that specifically target recall improvement. To this end, we can also add another agent to the system, for explicitly calling out entities the tagger agent failed to identify. Additionally, the system can be extended to incorporate a confidence scoring mechanism that allows agents to mark potential entities with varying degrees of certainty, rather than making binary decisions. This could help capture more valid entities while still maintaining the precision benefits of the multi-agent system.

6.2 More robust evaluation methods for generative NER

The development of AgenticNER exposed a significant gap in evaluation methodologies for generative NER systems. Our current heuristic method of handling modifications to input text by LLMs, while functional, makes a strong assumption about entity uniqueness within input sequences. This limitation affects evaluation accuracy in cases where identical token sequences can represent different entity types based on context. The problem is even more severe when input sequences are long and contain multiple sentences. Therefore, future work should focus on developing standardized evaluation methods specifically designed for generative NER systems. This can include new metrics that account for acceptable variations in the input and output sequences, new methods for converting the LLM output into sequence labeling ground truth format or new benchmarks specifically designed to evaluate NER as a generation task. The development of

better evaluation methodologies would not only benefit AgenticNER but also support the broader advancement of generative approaches to NER.

6.3 Better knowledge base integration

The varying effectiveness of the grounding engine across different benchmarks might be pointing to limitations in our current approach to knowledge base integration. Future research should explore more sophisticated methods for knowledge base utilization, possibly incorporating hierarchical relationships between entities and their attributes. Integration with domain-specific ontologies and knowledge graphs could provide richer context for entity verification and help reduce false positives while maintaining recall.

7 Conclusion

In this thesis, we introduced AgenticNER, a novel domain-agnostic NER framework that can adapt to any domain given only entity definitions and descriptions, without requiring domain-specific training data or prompt engineering. The framework utilizes collaborative multi-agent architecture where specialized agents - a tagger, a reviewer, and a researcher - work together to identify and verify named entities in text with a grounding engine providing additional verification by using external knowledge bases to validate entity candidates.

We evaluated AgenticNER across four distinct domains and demonstrated that the framework outperforms the single LLM call baseline in three out of four benchmarks, with precision improvements being most pronounced. However, we observed consistent challenges with recall performance, especially when using more efficient but less capable large language model.

The work on AgenticNER revealed three promising directions for future research. First, while our multi-agent architecture proved effective at improving precision, consistent challenges with recall performance suggest a need for research into techniques for recall improvement in collaborative multi-agent AI systems. Second, our experience with evaluating AgenticNER exposed a need for developing new robust evaluation methodologies for generative NER systems. Finally, the varying effectiveness of the grounding engine across different benchmarks pointed to opportunities for developing more sophisticated knowledge based grounding methods, potentially incorporating domain-specific ontologies and knowledge graphs.

AgenticNER represents a step toward more flexible and adaptable NER systems that can operate across domains with minimal setup requirements. While not a universal solution, it demonstrates the potential of collaborative multi-agent architecture in NLP tasks where domain expertise and careful reasoning are vital for success.

References

- [1] Brooke Cowan, Sven Zethelius, Brittany Luk, Teodora Baras, Prachi Ukarde, and Daodao Zhang. Named entity recognition in travel-related search queries. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29:3935–3941, Jan. 2015.
- [2] Yang GJ. Sohn CB. et al. Park, YJ. Gpdminer: a tool for extracting named entities and analyzing relations in biological literature. *BMC Bioinformatics* 25, 101, 2024.
- [3] Paweł Skórzewski, Mikołaj Pieniowski, and Grazyna Demenko. Named entity recognition to detect criminal texts on the web. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6223–6231, Marseille, France, June 2022. European Language Resources Association.
- [4] Marcel Gohsen and Benno Stein. Assisted knowledge graph authoring: Human-supervised knowledge graph construction from natural language. In *2024 ACM SIGIR Conference on Human Information Interaction and Retrieval*, pages 376–380, 03 2024.
- [5] Runyu Fan, Lizhe Wang, Jining Yan, Weijing Song, Yingqian Zhu, and Xiaodao Chen. Deep learning-based named entity recognition and knowledge graph construction for geological hazards. *ISPRS International Journal of Geo-Information*, 9(1), 2020.
- [6] Nitisha Jain. Domain-specific knowledge graph construction for semantic analysis. In Andreas Harth, Valentina Presutti, Raphaël Troncy, Maribel Acosta, Axel Polleres, Javier D. Fernández, Josiane Xavier Parreira, Olaf Hartig, Katja Hose, and Michael Cochez, editors, *The Semantic Web: ESWC 2020 Satellite Events*, pages 250–260, Cham, 2020. Springer International Publishing.
- [7] Miguel Menezes, Vera Cabarrao, Pedro Mota, Helena Moniz, and Alon Lavie. A case study on the importance of named entities in a machine translation pipeline for customer support content. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 211–219, Ghent, Belgium, June 2022. European Association for Machine Translation.
- [8] Arata Ugawa, Akihiro Tamura, Takashi Ninomiya, Hiroya Takamura, and Manabu Okumura. Neural machine translation incorporating named entity. In Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th*

- International Conference on Computational Linguistics*, pages 3240–3250, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.
- [10] Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. Recent advances in named entity recognition: A comprehensive survey and comparative study, 2024.
- [11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [12] Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. Gpt-ner: Named entity recognition via large language models, 2023.
- [13] Dhananjay Ashok and Zachary C. Lipton. Promptner: Prompting for named entity recognition, 2023.
- [14] Md. Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. MapCoder: Multi-agent code generation for competitive problem solving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4912–4944, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [15] Dong Huang, Jie M. Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation, 2024.
- [16] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024.
- [17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

- [18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.
- [19] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.
- [20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.
- [21] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA, 2024. Curran Associates Inc.
- [22] Miles Brundage Steven Adler Cullen O’Keefe Rosie Campbell Teddy Lee Pamela Mishkin Tyna Eloundou Alan Hickey Katarina Slama Lama Ahmad Paul McMILLAN Andrea Vallone Alexandre Passos David G. Robinson Yonadav Shavit, Sandhini Agarwal. Practices for governing agentic ai systems, 2023.
- [23] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. Survey Certification.
- [24] Barbara Von Eckardt. *What Is Cognitive Science?* The MIT Press, 1995.
- [25] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629, 2022.
- [26] Meta AI. Introducing llama 3.1: Our most capable models to date, 2024. <https://ai.meta.com/blog/meta-llama-3-1>.
- [27] Anthropic. Anthropic claude tool use documentation, 2024. <https://docs.anthropic.com/en/docs/build-with-claude/tool-use>.
- [28] OpenAI. Openai function calling documentation, 2024. <https://platform.openai.com/docs/guides/function-calling>.
- [29] Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

- [30] Patrick 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. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA, 2020. Curran Associates Inc.
- [31] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023.
- [32] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, pages i180–i182, 2003.
- [33] Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In *Conference on Empirical Methods in Natural Language Processing*, 2024.
- [34] Seoyeon Kim, Kwangwook Seo, Hyungjoo Chae, Jinyoung Yeo, and Dongha Lee. VerifNER: Verification-augmented NER via knowledge-grounded reasoning with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2441–2461, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [35] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710, 1965.
- [36] Andrea Zugarini, Andrew Zamai, Marco Ernandes, and Leonardo Rigutini. BUSTER: a “BUSINESS transaction entity recognition” dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 605–611, Singapore, December 2023. Association for Computational Linguistics.
- [37] Julia Evans, Sameer Sadruddin, and Jennifer D’Souza. Astro-ner – astronomy named entity recognition: Is gpt a good domain expert annotator?, 2024.
- [38] Elena V. Epure and Romain Hennequin. A human subject study of named entity recognition (ner) in conversational music recommendation queries. The 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023), 2023.

- [39] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- [40] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
- [41] Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. Evaluating large language models in class-level code generation. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24*, New York, NY, USA, 2024. Association for Computing Machinery.
- [42] Anthropic. Model card addendum: Claude 3.5 haiku and upgraded claude 3.5 sonnet, 2024.
- [43] Berk Atıl, Alexa Chittams, Liseng Fu, Ferhan Ture, Lixinyu Xu, and Breck Baldwin. Llm stability: A detailed analysis with some surprises, 2024.
- [44] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning, ICML'23*. JMLR.org, 2023.
- [45] Alexander Bondarenko and Adrian Viehweger. Llm robustness against misinformation in biomedical question answering, 2024.

Appendix

I. Source code

Source code for AgenticNER and evaluation experiments can be found at <https://github.com/narasul/agentic-ner>.

II. Example multi-agent conversation in AgenticNER

User

<text_to_tag>

Natural inflation from 5D SUGRA and low reheat temperature

</text_to_tag>

Tagger agent

<output>

<ResearchProblem>Natural inflation from 5D SUGRA</ResearchProblem> and

<Measurement>low reheat temperature</Measurement>

</output>

The reasoning for the tagging:

1. "Natural inflation from 5D SUGRA" is tagged as a <ResearchProblem> because it describes the main research investigation theme. - 5D SUGRA likely refers to 5-dimensional Supergravity, which is a research approach
2. "low reheat temperature" is tagged as a <Measurement> since it represents a specific physical property or parameter being studied
3. Following the tagging rules of prioritizing ResearchProblem and considering the context of cosmological research, these seem the most appropriate entity types.

Reviewer agent

Let me analyze the tagged entities:

1. <ResearchProblem>Natural inflation from 5D SUGRA</ResearchProblem>
 - "Natural inflation" is indeed a research problem in cosmology
 - "5D SUGRA" (5-dimensional Supergravity) seems more like a <Method> rather than part of the research problem
 - The method (5D SUGRA) is being used to investigate natural inflation
2. <Measurement>low reheat temperature</Measurement>
 - This could be a <Measurement>, but it might also be a <PhysicalQuantity>
 - "Reheat temperature" is a specific physical parameter in early universe cosmology

Let me ask for clarification:

<search>

In cosmology, is "5D SUGRA" more accurately classified as a Method or part of a ResearchProblem? Is "low reheat temperature" better classified as a Measurement or PhysicalQuantity?

</search>

I will await the researcher's input to provide a definitive assessment of the NER agent's tagging.

Research agent

I am searching for the following queries using the search tool:

```
{
  'queries': [
    '5D SUGRA in cosmology research method',
    'Reheat temperature definition in cosmology',
    'Classification of research methods in theoretical physics'
  ]
}
```

.... after getting response from search tool....

Based on the search results:

<answer>

1. 5D SUGRA is more accurately classified as a Method in cosmological research, specifically a theoretical approach for investigating natural inflation and related phenomena.
2. "Low reheat temperature" is better classified as a PhysicalQuantity, as it represents a specific temperature parameter in the cosmological reheating process that marks the transition from the inflationary epoch to the standard Big Bang model.

</answer>

Reviewer agent

Based on the researcher's clarification, I will provide feedback to the NER agent:

<feedback>

1. "5D SUGRA" should be tagged as <Method> instead of being part of the <ResearchProblem>
2. "low reheat temperature" should be tagged as <PhysicalQuantity> rather than <Measurement>

The corrected tagging should look like:

```
<ResearchProblem>Natural inflation</ResearchProblem> from <Method>5D
SUGRA</Method> and <PhysicalQuantity>low reheat temperature</PhysicalQuantity>
</feedback>
```

Tagger agent

```
<output>
<ResearchProblem>Natural inflation</ResearchProblem> from <Method>5D
SUGRA</Method> and <PhysicalQuantity>low reheat temperature</PhysicalQuantity>
</output>
```

I appreciate the detailed feedback and clarification. The revised tagging reflects the more precise categorization:

1. "Natural inflation" is now correctly isolated as the <ResearchProblem>
2. "5D SUGRA" is tagged as <Method>, recognizing it as a theoretical approach
3. "low reheat temperature" is now tagged as <PhysicalQuantity>, acknowledging its nature as a specific physical parameter in cosmological research

The changes improve the accuracy of the entity recognition by more precisely capturing the roles of each term in the scientific context.

Reviewer agent

Let me review the updated tagging:

1. <ResearchProblem>Natural inflation</ResearchProblem>
 - Correctly identifies the core research problem in cosmology
 - Focuses specifically on the research theme without including the method
2. <Method>5D SUGRA</Method>
 - Accurately represents 5-dimensional Supergravity as a theoretical approach
 - Properly classified as a method used to investigate the research problem
3. <PhysicalQuantity>low reheat temperature</PhysicalQuantity>
 - Correctly tagged as a physical parameter
 - Represents a specific quantitative aspect of the cosmological process

The revised tagging addresses the previous concerns and provides a more precise and nuanced representation of the entities in the text.

APPROVED!

Grounding engine

```
<grounding_feedback>
- 'natural inflation' is tagged as 'ResearchProblem'. It should likely be Process
instead.
</grounding_feedback>
```

Tagger agent

I'll carefully consider the grounding feedback about the classification of 'natural inflation'.

<output>

<Process>Natural inflation</Process> from <Method>5D SUGRA</Method> and

<PhysicalQuantity>low reheat temperature</PhysicalQuantity>

</output>

Reasoning for the change:

1. The grounding feedback suggests that 'natural inflation' is more accurately classified as a <Process> rather than a <ResearchProblem>
2. Natural inflation is indeed a physical process describing a specific mechanism of cosmic inflation
3. This aligns with the definition of Process as "Phenomena or associated processes"
4. The other tags remain the same as in the previous approved version

The modification reflects a more precise categorization of the term based on its fundamental nature as a cosmological process rather than a research problem.

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Rasul Nabiyev**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Collaborative Multi-Agent Architecture for Domain-Agnostic Named Entity Recognition,

(title of thesis)

supervised by Hendrik Šuvalov and Karl-Oskar Masing
(supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Rasul Nabiyev
08/01/2025