Lehrstuhl für Data Science

Species Interactions Network discovery using LLM across Scientific Papers

Masterarbeit von

Mahshid Ahmadi

1. Prüfer	2. Prüfer
Prof. Dr. Michael Granitzer	Prof. Dr. Annette Hautli-Janisz

July 8, 2025

Contents

Abstract

Understanding species interactions is central to biodiversity and ecological research, which are growing fields, producing valuable but complex data but hidden in text and documents. Retrieving this data manually is slow and costly, while automated methods would work better on large unstructured text. This problem is seen in databases like GloBI and Mangal.io, which have major limitations [Poi+21]. They are biased toward Western Europe and North America, with less data from Africa, South America, and Asia. Also, not all interaction types are equally covered, showing gaps in understanding global ecological patterns [Dan+24].

This thesis explores how large language models can help in extracting species interaction data from scientific literature to address this challenge. There is a two-step pipeline, the first step uses Named Entity Recognition to find species names. The second stage utilizes Relation Extraction to identify how species interact. The outcomes are used to create a knowledge graph that displays the connections among different species.

The best results were achieved using a strong LLM with few-shot prompting. In the NER stage, the model reached 91% recall, 44% precision, and a 60% F1-score. In the RE phase, it achieved 99% recall, 86% precision, and a 92% F1-score, with only a 15% error rate. Experts reviewed the extracted interactions, and the results were compared with two well-known ecological databases, GloBI and Mangal.io.

This work demonstrates that LLMs can support biodiversity research by automatically finding and organizing information about species interactions. The final knowledge graph presents new insights into ecological networks and enables researchers to better understand biodiversity on a global scale [Ant25a].

Acknowledgments

I would like to sincerely thank my supervisors, Prof. Dr. Michael Granitzer, Prof. Dr. Annette Hautli-Janisz, and Sahib Julka, for their guidance, support, and valuable feedback throughout this thesis. Their expertise and encouragement were essential to every step of this work.

I am also grateful to the Lehrstuhl für Data Science at the University of Passau for providing the academic environment and resources that made this research possible.

A special thanks goes to Dr. Babak Naeimi, whose knowledge and experience in biodiversity research were extremely helpful and inspiring during this project.

Finally, I would like to thank my family for their constant support, patience, and encouragement throughout this journey. Their belief in me made this work possible.

List of Figures

List of Tables

1 Introduction

Advances in artificial intelligence, especially Large Language Models (LLMs), have opened new possibilities for processing scientific text. These models can read large amounts of data and help answer questions automatically [Kom+24]. LLMs have been used in many fields to extract information and make the process faster and easier. Biodiversity science is one of these fields in which researchers are exploring LLMs to find the best approach to extracting information from ecological literature and building databases[Ant25c].

A big challenge is that much of the information on species interactions is trapped in written text. A lot of ecological knowledge is published in papers, PDFs, or reports, but not in structured databases. For example, GloBI (Global Biotic Interactions) is an effort to collect data on who-eats-whom and other interactions. It has gathered over 1.3 million interaction records from many sources [Mel+24]. Similarly, GBIF is the world's largest repository of species occurrence data, with over 2.3 billion records [MDA21]. Despite these huge databases, they still cover only a fraction of what is known. Most papers contain detailed observations about ecology, predator-prey links, plant-pollinator pairs, and more, but this information is not automatically accessible.

Researchers have built various tools and datasets to help with biodiversity information extraction. For example, named entity recognition (NER) has been used to find species and habitats in text. Datasets like BiodivNERE provide "gold-standard" corpora of biodiversity text labeled for NER and relation extraction (RE) tasks [Abd+22]. Domain-specific language models have also been created: BiodivBERT is the first BERT model pre-trained on biodiversity literature, and it showed improved performance on tasks like NER and RE in ecology [ALK23]. Other tools combine ontologies and NLP. For instance, there is a study that used a taxonomic NER tool (TaxoNERD) together with a fine-tuned LLM (LLaMA 2) to extract feeding relationships (predator–prey links) from

species descriptions [MRC24]. These approaches show that combining ontologies, NER, and relation extraction can discover interactions in text.

However, many of these efforts require specialized data or heavy annotation. They often focus on one language or group of organisms, or require manual evaluation. As an example, the Biodiversity Observations Miner finds co-occurrences of words to suggest interactions but still needs human verification [MKL19]. In summary, existing biodiversity informatics work provides useful components, but it has not fully automated the process of extracting the information yet.

Many biodiversity and species interaction datasets suffer from serious gaps and biases. Most recorded ecological networks come from temperate regions, while tropical areas like the Amazon and parts of Africa remain poorly covered. Interaction types are also unevenly studied, parasitism, mutualism, and predation each dominate in different climates, leaving entire ecosystems underrepresented [Poi+21]. Databases like GloBI and GBIF reflect these biases. GBIF, despite holding over a billion species records, shows strong spatial clustering around wealthy countries and research centers, which reduces the quality of species distribution models [Bec+14]. Studies show that relying on GBIF alone misses many species [DQR22]. These issues limit our ability to model ecosystems and make good conservation decisions.

To address these gaps, this thesis is going to answer:

1. How do Large Language Models (LLMs) extract and structure species interaction networks from Scientific literature in terms of recall, precision, and F1-Score?

2. What are the primary limitations of LLMs in identifying species interactions and ecological relationships, and how can these challenges be addressed to improve their applicability in biodiversity research?

These research questions will lead our study. Both prompt engineering and fine-tuning are going to be used as methods. The focus of the study is on LLAMA models.

To answer these questions, this thesis will test different LLM-based methods. We will use scientific texts, like abstracts from ecology papers, and create prompts to extract species names and their interaction types [OT+23]. We will also fine-tune an LLM using annotated examples from the Species-800 dataset, which contains 800 abstracts labeled with species names [Paf+13b]. This dataset helps the model learn how species are mentioned in real scientific writing. We will compare prompt-based and fine-tuned

models using standard measures like precision and recall. The main goal is to see if general LLMs can build species interaction networks from text. If successful, this can support better biodiversity research and help improve ecological databases [Ant].

2 Background

2.1 A Survey on Artificial Intelligence

Artificial Intelligence (AI) started as a research field in the 1950s. At that time, scientists wanted to build systems that could perform tasks similar to human thinking. The phrase "Artificial Intelligence" was first introduced at the Dartmouth Conference in 1956. In the following years, researchers developed rule-based systems called symbolic AI and expert systems [Mis24]. These systems worked well for some tasks but could not adapt to new situations [Ant25c].

In the 1970s and 1980s, progress in AI slowed down due to limited data and low computing power. This period was known as the "AI Winter" [RN16]. In the 2000s and 2010s, AI research grew again. Improvements in data storage, computer hardware, and learning algorithms have helped AI systems become more useful. For example, AI models like AlphaGo were able to learn complex games and perform better than humans [Sil+17].

AI includes many subfields. One of the most important is Machine Learning (ML). ML allows systems to learn from data and improve over time. There are three main types of ML: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning uses labeled data to train a model, such as using past house prices to predict future ones. Unsupervised learning finds patterns in data without labels, such as grouping customers based on shopping behavior. Reinforcement learning teaches a system through trial and error. The system receives rewards for good actions and penalties for bad ones. This method is used in robotics and games [Sil+17].

The figure below shows these types of learning:

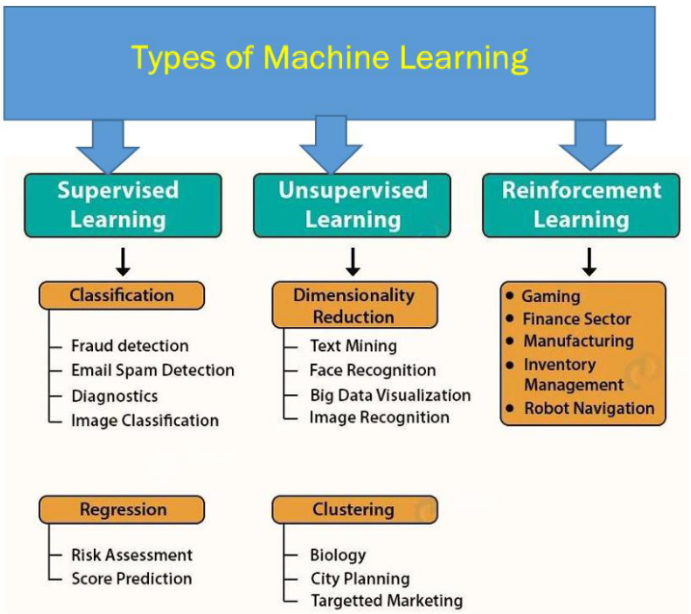

Figure 2.1: Types of Machine Learning [Mis24]

AI systems are built using different components. These include perception, reasoning, learning, natural language processing (NLP), and robotics. Perception allows machines to collect data through sensors or cameras. Reasoning helps machines make decisions based on logic. Learning lets machines improve their performance with experience. NLP helps machines understand and respond to human language. Robotics allows machines to move and perform tasks in the real world [BN06].

Deep learning is a type of machine learning that uses large neural networks. These networks are made of layers of connected units called neurons. Each layer processes the data and passes it to the next one. Deep learning works well for tasks such as image recognition and speech translation [Goo+16].

One important application of deep learning is Natural Language Processing (NLP). NLP is the field of AI that helps machines understand, process, and generate human language. It is used in systems like chatbots, language translators, and speech assistants. Deep learning models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have improved the way machines work with language by remembering sequences of words and understanding context [ZSV14]. These models are useful in tasks like machine translation, sentiment analysis, and question answering. As

AI systems become more advanced, NLP has become a key area in improving communication between humans and machines, especially in services such as customer support, voice commands, and virtual assistants [Vas+17b].

A major advancement in deep learning for NLP is the introduction of transformer models. Transformers use attention mechanisms to understand the relationships between words in a sentence, even when the words are far apart [Vas+17b]. This allows them to perform better than previous models in many language tasks. Models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) are examples of this new architecture [Chu+23]. These models are pre-trained on large datasets and then fine-tuned for specific tasks. This process, called transfer learning, saves time and improves performance [Vas+17b]. Transformers have become the standard in modern NLP and are used in applications like language translation, summarization, and text generation. Their ability to learn from large amounts of data and understand complex language patterns makes them one of the most powerful tools in AI today [Vas+17b].

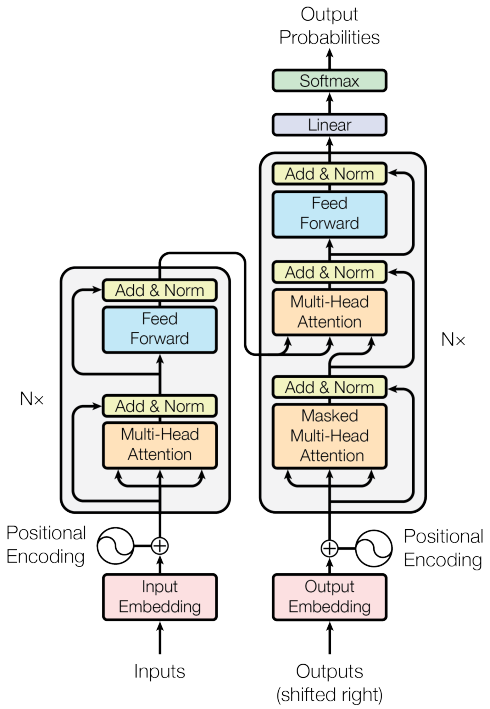

Figure 2.2: Architecture of the Transformer [Vas+17b]

2.2 Introduction to Large Language Models (LLMs)

Large Language Models (LLMs) are a type of artificial intelligence that focuses on understanding and generating human language. They use large sets of data to learn how to perform tasks. These tasks include answering questions, summarizing text, translating languages, and creating content [Had+23]. LLMs use machine learning techniques to learn patterns in text data, and they improve their performance by training on diverse language sources [Had+23]. While traditional AI was task-specific, LLMs offer wider adaptability across diverse domains [Had+23].

The key strength of LLMs is their ability to learn from data without requiring detailed human instructions. For example, after reading many examples, they learn grammar rules and facts and then apply them to new text [Zha+23]. One of the most famous LLMs is ChatGPT, created by OpenAI, which can have conversations and provide human-like responses. These models are based on the Generative Pre-trained Transformer framework and can understand context during long conversations [Ray23].

LLMs are part of a bigger family of models in natural language processing (NLP). Earlier models used rules or statistics, but LLMs use deep learning and transformer architectures [Dia+21]. These new methods allow the models to handle more complex tasks and understand language more deeply. LLMs are used in many areas, such as education, healthcare, law, finance, and customer service, showing their wide usefulness [Cho20]. As an instance, BloombergGPT is a trained model that supports a wide range of financial tasks and has outperformed other existing ones [Wu+23].

LLMs continue to improve as researchers find better training methods and larger datasets. Their development is supported by advances in computer hardware and by techniques such as fine-tuning and reinforcement learning with human feedback (RLHF) [YP24]. Also, training strategies like GrowLength, where the model begins with short input sequences and increases the length step by step, have been shown to accelerate convergence and improve performance without added complexity [Jin+23].

These improvements help LLMs give more accurate and safer outputs. As LLMs become more common, understanding how they work and using them properly is becoming an important topic in AI research and practice [Wei+22b].

2.2.1 Timeline and Historical Evolution of LLMs

The development of language models began with rule-based systems in the 1950s and 1960s, relying on manually constructed grammar rules for processing language [SBV98]. However, they were limited in scope and could not adapt to new or unexpected inputs. In the 1980s and 1990s, researchers began using statistical methods, such as n-grams and Hidden Markov Models (HMMs), to estimate the probability of word sequences based on observed frequencies [Ros00].

In the 2000s and early 2010s, machine learning models became more popular. These models learned from labeled datasets to perform specific NLP tasks like sentiment analysis or spam detection [NR13]. Around this time, Twitter and other platforms provided large text datasets that supported real-time language processing research [GHB09]. Machine learning models like Support Vector Machines (SVMs) brought improvements, but they still had limitations in understanding context and long-term dependencies in language [Joa98].

The next major shift came with deep learning. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models allowed machines to handle sequences of words and capture more context [Mik+10]. These models improved tasks such as speech recognition and machine translation. However, they had issues like vanishing gradients and difficulty processing long texts [Hoc98].

The transformer model was introduced in 2017 [Vas+17a]. This architecture employed self-attention mechanisms to understand all components of a sentence at once, which solved many problems of earlier models. Transformers enabled the training of significantly larger and more powerful models such as BERT [Dev+19] and GPT [GG20]. These advances set the stage for modern LLMs like GPT-3 and GPT-4, which can generate coherent and meaningful language at scale.

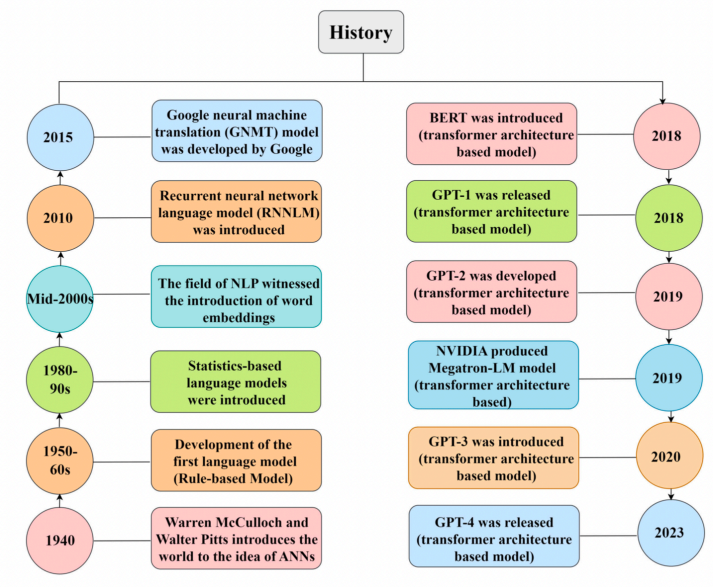

Figure 2.3: Timeline of Language Models [AI 23]

2.2.2 Fundamentals of LLM Architecture

LLMs are built on the transformer architecture. This design allows models to process an entire input sequence at once using a mechanism called self-attention [Dia+21]. Self-attention helps the model understand how each word in a sentence relates to every other word, even if they are far apart. This makes transformers much better than earlier models at capturing meaning and context in long sentences [Vas+17b].

Transformers are made of multiple layers. Each layer includes attention heads, feed-forward networks, normalization units, and residual connections [Vas+17b]. The attention heads let the model focus on different parts of the input. The feed-forward layers help transform and process the data. Residual connections and normalization ensure that the training process is stable and efficient [Dia+21].

Depending on the task, transformer models are used in different configurations. Encoder-only models, such as BERT and RoBERTa, are used for text understanding tasks like classification and named entity recognition [Liu+19]. Decoder-only models, such as GPT, are designed for text generation. Encoder–decoder models, like T5, are employed for sequence-to-sequence tasks such as translation and summarization [Lho+21].

Another important part of LLM architecture is positional encoding. Since transformers do not process text in order like RNNs, they need a way to understand word order. Positional encoding gives each word a unique signal based on its position, helping the model learn the structure of sentences [Vas+17b]. This combination of attention, layers, and position makes transformers very flexible and powerful for many tasks.

2.2.3 Generative AI and LLMs

Generative AI (GenAI) refers to AI systems that can create new content, such as text, images, code, or music. LLMs are a core part of GenAI because they can generate human-like language in response to a prompt [Bro+20]. This ability is used in many tools like ChatGPT, Bard, and CoPilot. These tools help people write, program, or answer questions quickly and easily [BLR23].

GenAI systems work by learning the patterns and relationships in the data they are trained on. During training, LLMs learn grammar, style, and meaning from large datasets [Bro+20]. Then, when given a prompt, they use that knowledge to generate responses that follow similar patterns. For example, they might complete a sentence, summarize a paragraph, or generate an email draft [Zen+21].

A typical GenAI system follows a design cycle with four main steps: define the task, select or build the model, fine-tune the model for specific tasks, and deploy it. Fine-tuning involves teaching the model to perform well in a particular application, such as medical advice or coding help. Deployment requires optimizing the model so it can respond quickly and reliably in real-world settings [Sch23].

LLMs use special prompting techniques to control how they generate content. Prompt engineering includes methods like zero-shot, one-shot, and few-shot prompting, where the model is given different types of instructions and examples [Whi+23]. These techniques make LLMs more useful and accurate for real tasks, and they are a key part of applying GenAI in practice [Koj+22].

2.2.4 LLAMA and its Pretraining Datasets

LLaMA (Large Language Model Meta AI) is an open-source LLM family developed by Meta. It is designed to be more efficient and accessible than some of the very large

models like GPT-4. LLaMA uses the transformer architecture and follows the same general training steps: pretraining on a large dataset and fine-tuning for specific tasks [Tou+23].

The LLaMA models are trained on diverse data sources. The datasets used for pretraining include CommonCrawl (3.3 TB), C4, GitHub, Wikipedia, books, ArXiv, and stock exchange data. These datasets cover many topics and writing styles, helping the model learn a wide range of knowledge and language usage [Tou+23].

Table 2.1: Pretraining datasets for LLaMA [Tou+23]

Dataset	**Sampling Proportion**	**Epochs**	**Disk Size**
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
GitHub	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
Stock Exchange	2.0%	1.03	78 GB

LLaMA is designed to be flexible and scalable. Smaller versions can run on personal computers or edge devices, making it more accessible for research and smaller organizations. Despite its smaller size, LLaMA achieves strong performance because of careful training and high-quality data [Tou+23].

By releasing LLaMA models as open-source, Meta has allowed the research community to explore and improve LLMs more easily. This encourages transparency and innovation in the field. Researchers can fine-tune LLaMA for their projects and contribute to improving its safety, performance, and fairness [Tou+23].

2.2.5 Variants of Transformer-Based LLMs

Transformer-based models come in many different forms, depending on how they are built and what tasks they are designed for. One popular variant is BERT (Bidirectional Encoder Representations from Transformers), which focuses on understanding language. It uses only the encoder part of the transformer to process input in both directions and is mainly used for tasks like classification, question answering, and text matching [Dev+19].

GPT (Generative Pre-trained Transformer), developed by OpenAI, is another important variant. GPT uses only the decoder part of the transformer and is designed for text generation. It reads input from left to right and is used in applications like chatbots and story generation. The GPT models have become more powerful over time, starting with GPT-1 (117M parameters) and growing to GPT-4 (1.76T parameters) [Ope+23].

Other transformer variants combine both encoder and decoder parts. For example, T5 (Text-to-Text Transfer Transformer) converts all tasks into a text-to-text format, making it highly flexible [Lho+21]. Models like XLNet and RoBERTa have also improved the performance and training strategies of BERT by changing the order of word prediction or increasing data size [Liu+19].

In addition to these general-purpose models, some transformers are built for specific domains. For example, BioBERT and SciBERT are trained on scientific or biomedical texts, and CodeX is designed for programming tasks. These variants show how transformer-based LLMs can be adapted to different fields by adjusting their training data and task objectives [Xu+22].

2.2.6 State-of-the-Art LLMs

From 2019 to 2024, the field of LLMs has grown rapidly. In 2020, GPT-3 set a new benchmark with 175 billion parameters and strong few-shot learning abilities. It could generate high-quality text with only a few examples [Bro+20]. Soon after, many other models followed with different focuses, including T5 (Google) [Raf+20], ERNIE (Baidu) [Zha+19], and Jurassic-1 (AI21 Labs) [Lie+21].

In 2022 and 2023, more advanced models like Chinchilla (DeepMind), GLaM (Google), and PaLM (Google AI) focused on scaling efficiency. Chinchilla showed that increasing the amount of training data can be more effective than increasing model size alone [Hof+22]. GLaM introduced a ”mixture of experts” technique to reduce computation while maintaining high performance [Du+22].

In 2023, GPT-4 was released with even stronger abilities in reasoning, creativity, and multimodal inputs (text and image) [Ope+23]. Other competitive models included Claude Instant (Anthropic), LLaMA 2 (Meta) [Tou+23], and Falcon (TII) [Pen+23]. These models differ in size, cost, speed, and target applications, showing that the LLM space is now both competitive and diverse.

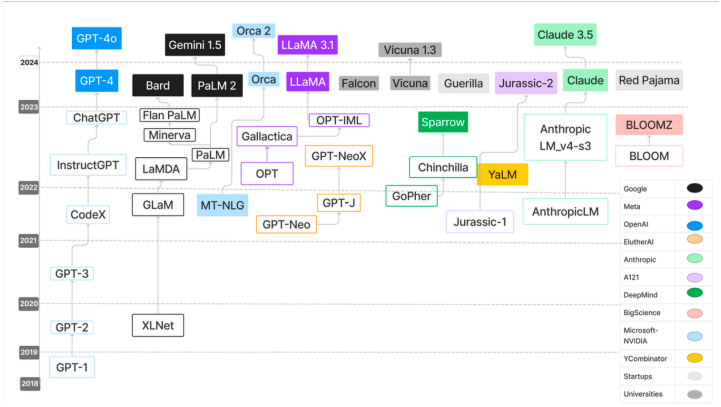

Figure 2.4: Timeline of LLMs from 2019-2024 [Had+23]

2.2.7 Structure and Training of LLMs (including Fine-tuning)

LLMs are usually trained in two main stages: pretraining and fine-tuning. Pretraining uses large, general datasets from the internet, books, or code to teach the model grammar, facts, and language structure [Zha+23]. This phase gives the model general knowledge and the ability to understand many types of text.

Fine-tuning is the next stage. It adjusts the pretrained model to perform specific tasks, such as customer support, summarization, or medical diagnosis. This is done by training the model on a smaller, domain-specific dataset. Fine-tuning helps improve accuracy and relevance for the target application [YP24].

During training, techniques like Reinforcement Learning with Human Feedback (RLHF) are used to guide the model toward safe and helpful responses [Bai+22]. RLHF involves human evaluators ranking model outputs, which helps train the model to avoid toxic, biased, or factually incorrect responses [Ouy+22].

The structure of LLMs includes many layers of self-attention, feed-forward networks, and normalization. Training such models requires a large number of GPUs and high memory. As a result, research is also focusing on optimizing training strategies and reducing energy use without harming performance [Cha23].

2.2.8 Prompt Engineering on LLMs

Prompt engineering is the process of designing effective inputs (prompts) for LLMs to get the best possible results. Prompts act like instructions that guide the model's output. For example, writing "Translate this to French: Hello" helps the model understand the task. Prompt engineering has an important role in making LLMs useful in real-world applications [Whi+23]. Prompting is also used in training, especially in fine-tuning and in-context learning. Users can give examples inside the prompt instead of updating the model weights. This allows quick adaptation to new tasks and is especially helpful in situations with limited training data [Hu+23]. There are several prompting techniques. These techniques help the model understand the task and generate more accurate responses [Dan+22].

- Zero-Shot Prompting is giving a model an instruction in natural language without any further examples. The model uses only its pre-trained knowledge to generate the answer. For example, to extract named entities from a sentence, a zero-shot prompt could be "Extract all the people and locations in this sentence: [text]". This method is simple and fast, and it is helpful when no labeled examples are available [Sah+24]. However, zero-shot may perform poorly on more complex problems such as relation extraction or fine-grained entity typing. This is because the model does not receive any specific examples to follow. As shown in domain-specific tasks, the zero-shot accuracy is often lower than example-included prompting methods [Zha+25].

  Zero-shot Input Prompt

  Classify whether the sentence below is a Positive, Negative, or Neutral one. No explanation for your choice.
  Text: {text}
  Sentiment:

Zero-shot Output Examples

Text: The movie was great.
Sentiment: Positive

Text: The weather is typical for this time of year.
Sentiment: Neutral

Text: The ice cream is melted.
Sentiment: Negative

- Few-shot prompting adds some examples in the prompt to help the model understand the pattern. This method was popularized by GPT-3, where a small number of demonstrations are placed before the main input [Bro+20]. A few-shot prompt might show two or three sentences with entity tags, followed by a new sentence that the model is expected to label similarly.

Input of Few-shot Prompting

Classify the sentences as **Positive**, **Negative**, or **Neutral**.
Examples:
Text: This class is helpful.
Sentiment: Positive

Text: THe concert was not satisfying.
Sentiment: Negative

Text: The weather today is okay.
Sentiment: Neutral

Now, classify the following:
Text: {input_text}
Sentiment:

Output

Input: The food of the restaurant was unbelievably delicious!
Predicted Sentiment: Positive

Few-shot prompting is effective in low-data settings and works well when the example format is clear. However, it can be sensitive to the order and choice of

examples and input length [Sah+24]. In NER tasks, this method often improves performance compared to zero-shot setups, but still it is not better than fine-tuned models in terms of accuracy [AL23].

- Chain-of-Thought Prompting (CoT) is a technique where the prompt enables the model to reason step-by-step. Instead of giving a short answer, the model is asked to show the reasoning process first and then give the final answer [Wei+22a]. Chain-of-thought prompting has been especially useful in tasks requiring multi-step reasoning or commonsense logic [Wei+22a]. It has been shown to significantly improve the performance of models like PaLM and GPT-3 on benchmarks such as GSM8K and AQuA [Sah+24]. In NER and relation extraction, this method can be adapted by asking the model to first locate candidate spans and then reason about their types [Sah+24].

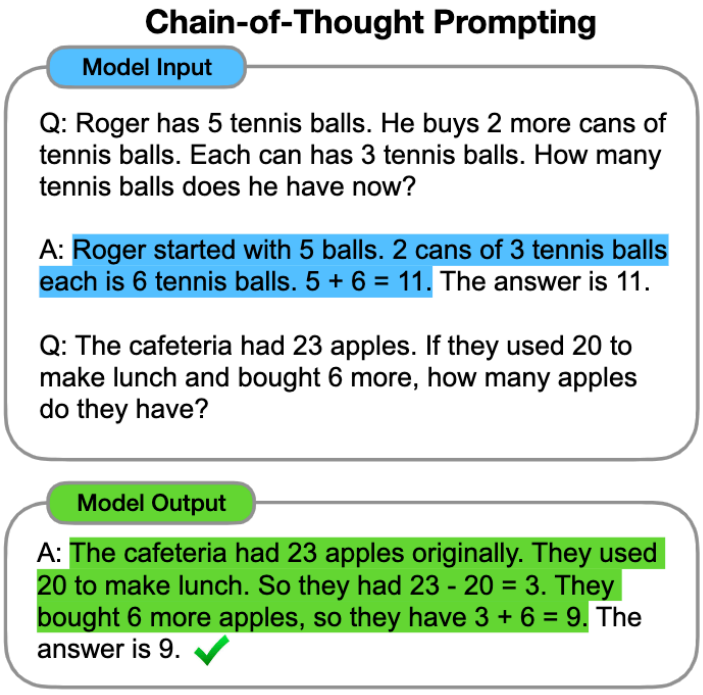

Figure 2.5: Example of COT prompting [Wei+22a]

2.2.9 Comparison: ML, DL, and LLMs

Traditional machine learning (ML) models use structured data and hand-engineered features to solve specific tasks like spam detection or image classification. They require labeled datasets and domain knowledge to design good input features [KZP06]. ML models include decision trees, support vector machines, and Naive Bayes classifiers [Cer+20].

Deep learning (DL) introduced the use of neural networks to learn patterns directly from data. Models like CNNs and RNNs work well for images and sequences and can learn more complex patterns than ML[Yin+17]. DL models are used in tasks like speech recognition and computer vision [Raw+22].

LLMs build on DL by using transformers and much larger datasets. They can perform many tasks without retraining for each one. Unlike ML and DL models, which often require separate models for each task, one LLM can handle multiple tasks using prompting or minor fine-tuning [Ope+23].

Another key difference is scalability. ML models do not improve much with more data. DL improves with data but still needs separate models for different tasks. LLMs continue to improve with more data and parameters and can generalize better [Kim+21]. This makes them more powerful but also more resource-intensive [Cha23].

Table 2.2: Comparing Traditional ML, Deep Learning, and Large Language Models [Had+23]

Factor	**ML**	**DL**	**LLMs**
Training Data Size	Medium	Large	Very Large
Feature Engineering	Required	Partial	Not Required
Model Complexity	Low	High	Very High
Interpretability	High	Low	Low
Performance	Moderate	High	Very High
Hardware Requirements	Standard	High	Very High
Scalability	Limited	Moderate	High
Fine-tuning	Not Applicable	Possible	Possible
Transfer Learning	Limited	Effective	Effective
Contextual Understanding	Limited	Moderate	High

2.2.10 Applications of LLMs

LLMs are used in many areas, such as education, healthcare, law, and customer support. In education, they can act as tutors, generate learning materials, or answer student questions [Kas+23]. In healthcare, they help with medical summaries and clinical predictions [Sal23], and even with scientific applications like protein modeling [Lin+22].

In finance, LLMs are used for fraud detection, risk assessment, and report generation. They help in automating repetitive tasks and analyzing large volumes of financial text data [Wu+23]. Legal professionals use LLMs to summarize documents, draft contracts, or assist with legal research [Sun23].

In software development, models like GitHub Copilot or CodeGen help developers write code, find bugs, or understand legacy code [Che+21]. In media and entertainment, LLMs assist in content creation, from writing scripts [Wan+24] to generating news summaries [MV22].

With the addition of multimodal capabilities in models like GPT-4, applications now include vision-language tasks such as image captioning, visual QA, and document reading. This opens up new use cases in medical imaging, document automation, and even art generation [Hua+23].

2.3 Introduction to Species Interaction Network

Scientists use interaction networks to show how species in a community connect with each other. In these networks, each species is shown as a node, and the interaction between them is a link [Váz+22].

For example, one network might show which insects pollinate which plants, while another might show who eats whom. These connections form a web of interactions.

This method includes different types of interactions [PSM14]:

Table 2.3: Species Interaction Relations and Their Directionality

Relation	**Source**	**Target**
eats	consumer	food

Relation	Source	Target
eatenBy	food	consumer
preysOn	predator	prey
preyedUponBy	prey	predator
kills	killer	victim
killedBy	victim	killer
parasiteOf	parasite	host
hasParasite	host	parasite
endoparasiteOf	endoparasite	host
hasEndoparasite	host	endoparasite
ectoparasiteOf	ectoparasite	host
hasEctoparasite	host	ectoparasite
parasitoidOf	parasitoid	host
hasParasitoid	host	parasitoid
hostOf	host	symbiont
hasHost	symbiont	host
pollinates	pollinator	plant
pollinatedBy	plant	pollinator
pathogenOf	pathogen	host
allelopathOf	pathogen	host
hasPathogen	host	pathogen
vectorOf	vector	pathogen
hasVector	pathogen	vector
dispersalVectorOf	vector	seed
hasDispersalVector	seed	vector
hasHabitat	inhabitant	habitat
createsHabitatFor	habitat	inhabitant
epiphyteOf	plant/algae	host plant
hasEpiphyte	plant	plant/algae
providesNutrientsFor	host	consumer
acquiresNutrientsFrom	consumer	host
symbiontOf	symbiont	symbiont
mutualistOf	mutualist	mutualist
commensalistOf	commensalist	commensalist
flowersVisitedBy	plant	visitor

Relation	Source	Target
visitsFlowersOf	visitor	plant
ecologicallyRelatedTo	source	target
coOccursWith	source	target
coRoostsWith	source	target
interactsWith	source	target
adjacentTo	source	target

Understanding these networks helps scientists see how ecosystems work and what may happen if some species disappear. The GloBI project [PSM14] adds to this by making millions of interactions available in a shared system, so researchers can find answers more easily, like “What do sharks eat in California?”

To build a network, scientists collect data by observing interactions between species. Often, the networks are bipartite, with two groups, like plants and insects. A link is created when one interacts with another, and the strength of this link can be measured by how often it happens [VA05]. GloBI makes this process easier by collecting and connecting data from many sources into one big, machine-readable dataset. It uses common systems like taxonomies (like NCBI, ITIS) and ontologies (such as EnvO, Uberon) to keep everything organized [PSM14]. Figure 2.6 shows how an interaction is recorded with information about species, location, and environment.

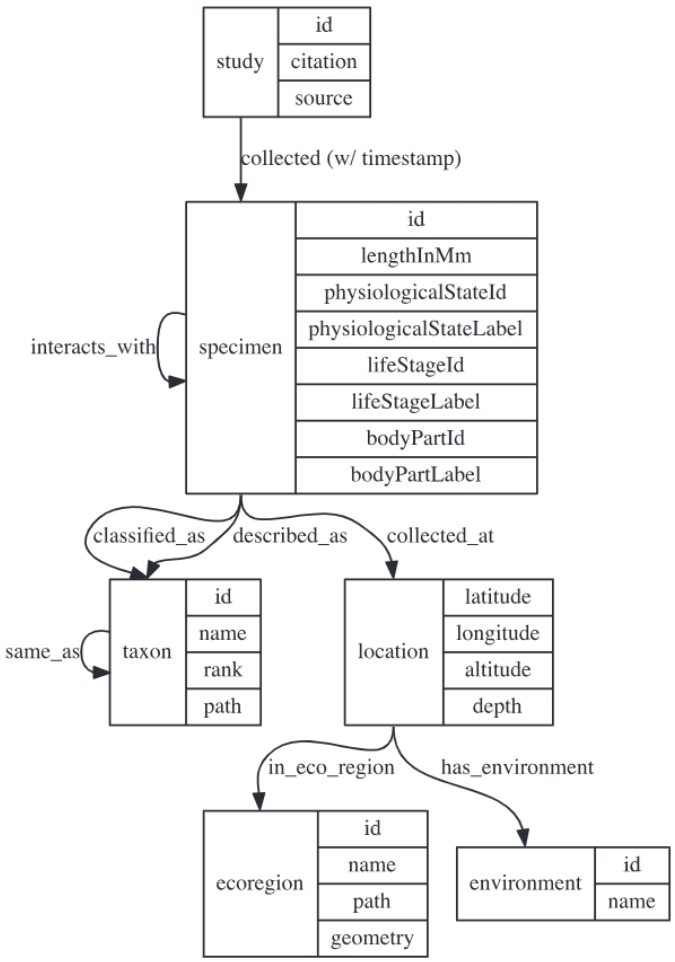

Figure 2.6: Structure of the Interaction Dataset [PSM14]

Interaction networks show special patterns. One is nestedness, where specialist species (with few links) interact with generalist species (with many links) [Bas+03]. Another is modularity, where species form groups that interact more within the group than outside [Ole+07]. These patterns help the ecosystem stay stable. For example, nestedness allows for more flexibility, so if a species is lost, the network still works. Modularity keeps problems, like disease, within small parts of the network.

These networks can be affected by climate change, habitat loss, and poor sampling. When sampling is weak, rare or weak interactions might be missed, known as the "lazy sampling effect" [VA05]. Climate change may shift flowering seasons, so some pollinators cannot find food in time [BA11]. GloBI helps by combining over 700,000 interactions

across 50,000 species from more than 1,100 studies. This database has high coverage in the North Sea, the Gulf of Mexico, and the Southern Ocean. [PSM14]

Species Interaction Networks are important for protecting nature and the wild environment. Some species act as hubs in the network, which are connected and hold the structure together. If one of them disappears, it may break the entire network into isolated parts [GDO10]. Tools like GloBI and platforms like the Encyclopedia of Life (EOL) help by collecting, organizing, and sharing this valuable information. With better access to such networks, scientists can understand which species are most important and how to protect ecosystems from collapse.

2.4 Named Entity Recognition and Relation Extraction

Named Entity Recognition (NER) and Relation Extraction (RE) are two key areas in Natural Language Processing (NLP) and Information Extraction (IE). Their goal is to convert unstructured textual data into structured and usable knowledge. These methods are especially important in fields like biomedicine and ecology, where vast amounts of research literature contain valuable information that is difficult to process manually. Historically, early approaches to NER and RE were based on manually written rules and dictionaries, commonly used in tasks like recognizing names and organizations in news texts. Over time, statistical models such as Conditional Random Fields (CRF) and Support Vector Machines (SVM) replaced rule-based methods. These models used engineered features to learn how to recognize entities and relationships from labeled data. The recent shift toward deep learning has brought significant improvements, especially with models like BiLSTM-CRF [Hab+17] and transformer-based systems such as BERT, which can automatically learn useful patterns from large datasets without extensive manual feature design [Dev+19].

NER focuses on identifying and categorizing important terms or concepts mentioned in text. In specialized fields such as biomedical research, this includes detecting mentions of genes, diseases, proteins, or species. For example, a system might recognize ”TP53” as a gene or ”SARS-CoV-2” as a virus. In this context, the task is often referred to as BioNER. Accurate NER in such domains is challenging due to factors like unclear boundaries between entities, frequent use of abbreviations, and the presence of synonyms

for the same term [YB19]. Moreover, new entities are constantly introduced as scientific discoveries progress, which adds to the complexity of the task [GS24].

Relation Extraction, on the other hand, involves identifying the type of relationship that exists between two or more entities in a sentence or document. These relationships can describe actions, associations, or interactions. In biomedicine, RE is used to detect whether a drug treats a disease or whether a gene is linked to a specific condition. In ecological texts, RE can uncover relationships such as competition or predation between species. However, RE systems face several difficulties. These include inconsistent naming of entities across texts, the presence of multiple types of relationships between the same entities, and the fact that most entity pairs in a document do not have a meaningful relationship. This imbalance between related and unrelated pairs can negatively affect model performance [Li+23a].

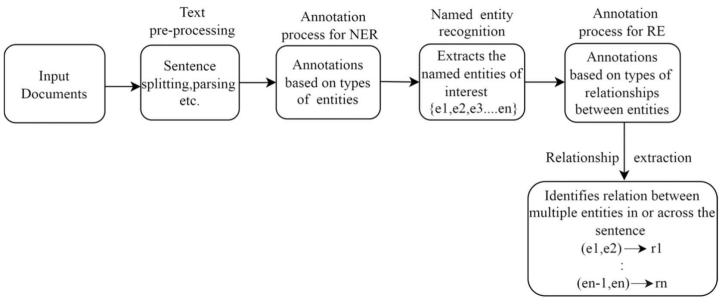

Figure 2.7: The connection between NER and RE [GS24]

Recent advancements in NER and RE have been driven by the use of deep learning methods. Neural network models such as BiLSTM combined with CRF layers have become widely used for NER due to their ability to capture both short and long context in sequences of text. In the field of RE, neural architectures like convolutional neural networks (CNNs) and attention-based LSTMs are commonly used to detect complex relationships in text. A major development has been the introduction of pre-trained transformer models like BERT, which learn general language patterns from large corpora and can then be fine-tuned for specific tasks such as biomedical NER and RE. BioBERT, a domain-specific variant of BERT, has shown strong results on several biomedical datasets and is now a standard baseline for many biomedical text mining tasks [Lee+20].

In cases where computational resources are limited, researchers have proposed lightweight and cost-effective models. One example is BINER, which was designed to operate effi-

ciently even on low-end machines. Such models are helpful when deploying NLP tools in real-world applications without access to powerful hardware or cloud services [ASE22].

Another important trend is the move toward joint learning models. Instead of separating NER and RE into two steps, some systems train both tasks together. This integrated approach avoids the problem of error propagation, where mistakes in entity recognition lead to incorrect relation predictions [Yad+20], and improves overall performance by learning shared representations for both tasks [Bek+18].

Additionally, hybrid systems that combine rules, dictionaries, and deep learning have been explored to improve robustness. For example, one system used a BiLSTM-CRF model with negation-based transfer learning to improve recognition of medical entities and their relationships in clinical text [Fab+23].

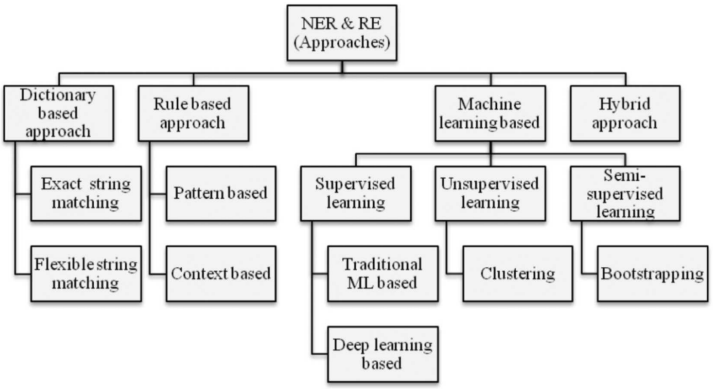

Figure 2.8: Different Approaches in NER and RE models [GS24]

NER and RE are widely applied across various scientific and industrial domains. In biomedical research, they support drug discovery by identifying associations between genes, diseases, and compounds. They also assist in pharmacovigilance by detecting mentions of adverse drug reactions in clinical notes and scientific literature [GS24].

In ecological and biodiversity studies, NER and RE are used to extract species names and their interactions from environmental papers, allowing researchers to build species interaction networks and monitor ecological dynamics [Fab+23].

In the clinical field, these techniques help process electronic health records (EHRs) to extract symptoms, diagnoses, treatments, and other critical information that can support medical decision-making and improve patient care [Yad+20].

Beyond science and medicine, NER and RE also contribute to large-scale literature mining and knowledge graph construction, enabling faster and more efficient access to research findings [GS24].

The development of Named Entity Recognition and Relation Extraction has moved from simple rule-based systems to advanced deep learning models capable of handling large, complex datasets [YB19]. These methods are now central to extracting useful knowledge from biomedical and ecological texts [Lee+20]. While transformer models and joint learning approaches have improved the performance of NER and RE, several challenges remain. These include dealing with entity ambiguity, class imbalance, and the need for annotated data [Li+23a]. However, the applications of these techniques continue to expand, offering powerful tools for scientific discovery and real-world decision-making [GS24].

2.5 Knowledge Graph

Knowledge graphs (KGs) are a way to organize and connect information using a graph format. In a KG, real-world objects such as people, places, or concepts are shown as nodes, and the relationships between them are shown as edges. Each fact in a KG is stored as a triple: (head entity, relation, tail entity). For example, the triple (Albert Einstein, bornIn, Germany) links a person to a country using a specific relation. These graphs help machines understand how concepts are related and allow reasoning based on those connections [Nic+15]. Because of this, knowledge graphs are now used in many applications like search engines, recommendation systems, and scientific research [Jia+23].

The idea of KGs comes from early work in the Semantic Web, which used standards like RDF and OWL to describe knowledge [BHL01; AV04]. Projects such as Cyc and WordNet also helped define structured knowledge. However, the term “knowledge graph” became popular after Google introduced its Knowledge Graph in 2012 [Sin+12]. Since then, many other KGs have been created. These include DBpedia [Aue+07], Freebase [Bol+08], YAGO [SKW07], and Wikidata [VK14]. Some graphs are general-purpose, while others focus on specific areas like medicine or biology. These graphs are built from sources like Wikipedia, scientific texts, and databases.

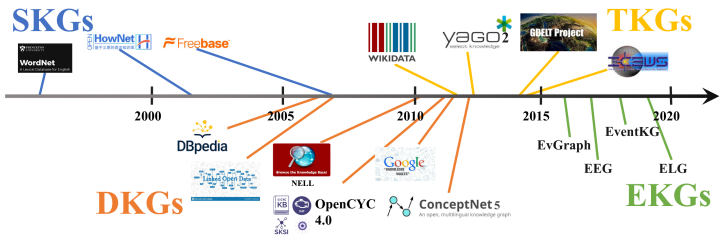

Figure 2.9: Different Approaches in NER and RE models [Jia+23]

To build a knowledge graph, there are three main steps: finding the entities, identifying how they are related, and linking them together into a single graph. Entity recognition is the task of finding important terms in the text, such as names of people, genes, or chemicals. Relation extraction finds how these entities are connected. Tools like BERT [Dev+19] and attention-based models [Vas+17b] help improve these tasks. Once the facts are extracted, entity linking is used to connect new information to existing entries and remove duplicates. This process ensures that the graph is complete and well-organized [Jia+23].

After a KG is built, it can be improved by adding missing information or predicting new relationships. This process is called knowledge completion. A popular way to do this is through knowledge graph embeddings (KGE), which convert entities and relations into mathematical vectors. These vectors help in tasks like predicting new links, classifying entities, and finding similar concepts. Well-known embedding models include TransE [Bor+13], DistMult [Yan+14], ComplEx [Tro+16], and RotatE [Sun+19]. These models make it easier for computers to work with large and complex graphs.

KGs are used in many real-world applications. In web search, they help answer questions and suggest related topics [Don+14]. In recommendation systems, they suggest products or content based on a user's interests by modeling connections in the graph [Wan+18; Wan+19]. In medicine, knowledge graphs like Hetionet [Him+17] and DRKG [IZK20] connect drugs, diseases, and genes to support research and drug discovery. In science, they help organize and search through thousands of research papers, making it easier for scientists to find useful information [Jia+23].

Even though KGs are very useful, there are still some challenges. These include missing or wrong information, changes over time, and the need to combine information from different types of data, like images, text, and tables. Another problem is making sure

that knowledge graph systems are fair, clear, and easy to understand. New research is trying to solve these problems using hybrid models that mix rule-based reasoning with deep learning [Bes+21]. Also, combining large language models with KGs may lead to better knowledge extraction and reasoning in the future [Jia+23].

3 Methods

This project includes different stages and implementations. The codebase is developed in Python v. 3.11.7 and relies on NumPy, Pandas, Torch, Sklearn, BeautifulSoup, Transformers, Langchain, and NetworkX either directly or indirectly. As mentioned, this thesis aims to automatically extract and integrate species interaction networks from text-based sources, such as scientific literature [Ant25d].

3.1 Dataset

The Species-800 (S800) dataset is a well-known gold-standard dataset used for species name recognition in scientific texts [Paf+13a]. It was developed to support research in Named Entity Recognition, especially for identifying species in biomedical and biodiversity literature. Unlike regular NER datasets that focus on people, organizations, or locations, this dataset focuses only on species names like Homo sapiens or Escherichia coli. These names are often written in complex scientific ways, making them harder to recognize using traditional NER tools. The dataset is used for evaluating how well a model can detect species names in sentences and tag them correctly. Every species name in the dataset is marked using the BIO tagging format, which stands for Beginning (B), Inside (I), and Outside (O) of named entities [Paf+13a].

The S800 dataset was created in 2013 as part of the work on the SPECIES and ORGANISMS tagging tools [Paf+13a]. The authors built this dataset because existing datasets, such as Linnaeus-100 [GNB10], had limitations like low species diversity and repeated mentions of the same names. To overcome these issues, the S800 corpus was built from 800 different abstracts taken from eight different fields: bacteriology, botany, entomology, medicine, mycology, protistology, virology, and zoology. For each of these

categories, 100 abstracts were selected, ensuring the dataset contained a wide range of species types. Five human curators manually annotated the abstracts, following strict guidelines. Each abstract was reviewed to highlight species names and make sure they were correctly labeled. To check the quality of the annotations, 20% of the abstracts were labeled by two curators. The average agreement between them was high (Cohen's kappa = 0.80), showing the dataset is reliable [Paf+13b].

The Species-800 dataset is divided into three main parts: a training set, a validation set, and a test set. The training set includes 5,734 sentences, the validation set has 831 sentences, and the test set contains 1,631 sentences. In total, there are 8,196 annotated sentences. These annotations follow the BIO scheme and cover many forms of species names. The dataset includes both full scientific names, like Panthera leo, and abbreviated names, such as C. sativa. Figure 3.1 shows the number of species across different categories. For example, virology had the most mentions (946), while medicine had the fewest (90). The dataset contains over 718 unique species and 3,708 total mentions. Because of its wide coverage, S800 is a good choice for testing how well models can handle the diversity in scientific naming conventions [Paf+13b][Néd+13].

Corpus	Category	Documents	Unique species	Unique names	Mentions
S800	Protistology	100	196	284	497
	Entomology	100	138	293	614
	Virology	100	117	342	946
	Bacteriology	100	87	179	416
	Zoology	100	85	160	299
	Mycology	100	80	178	538
	Botany	100	68	131	308
	Medicine	100	23	30	90
	Total	800	718	1503	3708

Figure 3.1: The Categories and Number of the species in Species800 [Paf+13b]

In this research, the Species-800 dataset is used for two main natural language processing tasks: Named Entity Recognition and Relation Extraction. The dataset provided the foundation for building a pipeline that could detect species names from scientific text and identify the interactions between them. Both few-shot learning methods and fine-tuning approaches on large language models are used to develop an automated system

that extracts species interactions from ecological literature.

3.2 Preprocessing and Data collection

As the first step towards this goal, a Python-based paper-downloader pipeline has been developed to locate and download any relevant papers related to biodiversity and ecology. The Paper-Downloader script has been developed to simplify the process of gathering academic papers from various major databases, making it easier and faster than manually gathering them. It operates through a primary Paper-Downloader class that manages all download operations.
The pipeline works as follows: First, gather queries consisting of keywords to search for relevant papers, such as [”ecological networks”, ”trophic interactions”, “network ecology”, ”food web structure”, ”species clustering”, ”predator-prey dynamics”, ”mutualistic interactions”, ”species”, AND ”interaction” AND “ecology”’]. This list can be extended to include more keywords, making the search and sources more specific. After obtaining the queries, it is necessary to determine the number of papers that need to be downloaded. With the queries and the desired number of papers, this data will be passed to the function that manages the scraping and downloading of the papers. This function uses three portals: Arxiv, Springer, and Elsevier. For each database integration, there is a different implementation method:

- The ArXiv method processes XML-based responses.
- The Springer method handles JSON data.
- The Elsevier method manages a more complex multi-step verification process.

Each method includes specific error handling and rate limiting appropriate to its respective platform. Data management occurs at two distinct levels within the script: managing paper downloads and organizing metadata records. First, the script manages paper downloads, generates appropriate filenames, and organizes PDFs into the set directory structure. Second, it keeps comprehensive metadata records for each paper, including titles, authors, publication dates, and abstracts. This implementation gives two separate outputs: one is the downloaded PDFs, and one is the metadata file of these PDFs. This metadata is stored in separate files in the source database, forming a

searchable record of all downloaded materials containing important information, including paper titles, authors, publication dates, abstracts, and source database details.
This is a sample of the produced metadata.

Title: Community perceptions and socio-economic implications of conservation corridors and networks in the Vhembe District, Limpopo, South Africa

Authors: Dalziel, Alexandra and Evans, Mary

Published: February 1, 2025

Source: Springer

Summary: Social facets linked to conservation corridors and ecological networks have received relatively limited academic attention. This study explores the perspectives of researchers, NGO representatives, and landowners, as well as the community's ideas of conservation efforts and corridor potential in the Vhembe District, Limpopo, South Africa. Surveys and interviews were conducted with communities, regional stakeholders, and landowners. The findings revealed that the community participants strongly support corridor implementation. The results indicate that this support is driven by the anticipated socio-economic benefits in the form of jobs. However, the employment opportunities might not align with the residents' expectations. The study identifies several challenges to corridor establishment, including infrastructure and financial constraints. Moreover, the findings revealed a lack of supportive legislation and highlighted concerns over the protected area's accessibility. The study contributes to the global academic discourse by emphasizing the importance of community engagement before corridor and network implementation. It also addresses the complex trade-offs inherent in such projects, regardless of location. The methodological approach employed in this research transcends its regional context and offers actionable insights applicable worldwide.

The outputs of the Paper-Downloader will be used as raw text and converted to cleaned and normalized text. This processed text is fed into large language models for Named Entity Recognition and Relationship Extraction to identify species and their interactions.

Using this system needs care regarding

- Authentication Needs: The script uses API keys for Springer and Elsevier.

- Rate Limiting: The script implements mandatory delays between requests (1-3 seconds) to comply with database usage policies.

Once the papers have been fetched, the next step is to extract the textual content of each paper to create a corpus for analysis and training the model. Removing unnecessary characters or words from this corpus and preparing it for use with our prompts is important. To meet this need, we employ a Python script to extract text from all the papers, regardless of how they are presented—whether in a one-column or two-column format—and store the contents in individual files. At the core of this process, there is a class called ScientificPaperExtractor, which processes different types of documents to extract text from them. The main package used in this implementation is PyMuPDF (or Fitz) to handle PDF documents, providing proper functionality to cross through complex PDF layouts. When we assemble the ScientificPaperExtractor, the extraction is performed page-wise, and for a single page, it is checked whether it is a single-column page or a double-column page, and then a proper function for text extraction is used. The script includes an error-handling feature, so any issue on a single page won't hinder the extraction of a whole document. Then, the output extracted text is stored in a file named the same as the source, but the extension is changed to a .txt file. This process is done by three essential functions that handle the technical aspects of text extraction:

1. is_two_column_layout(page): This function analyzes a page to determine whether it uses a single-column or double-column layout. It works by examining the x-coordinates of text blocks on the page and looking for a clear separation between the left and right columns. The page is considered to have a two-column layout if there are text blocks positioned in both the left and right halves of the page.

2. extract_text_from_single_column(page): This function handles text extraction for pages with a single-column layout. It uses PyMuPDF's built-in text extraction capability to process the entire page as a continuous text block [Ic24].

3. extract_text_from_two_columns(page): This function manages the more complex task of extracting text from double-column layouts. It first divides the page into left and right halves, then sorts text blocks in each column based on their vertical position to maintain proper reading order. Finally, it combines the text from both columns, ensuring the content remains in the correct sequence.

After extracting raw text from scientific papers, the system employs two specialized implementations to prepare the text for the model: the TextCleaner and CorpusBuilder

classes. These classes work consecutively to transform the raw extracted text into a well-structured, clean corpus that can be effectively used for the model.

The TextCleaner script is a Python-based tool designed to clean and preprocess raw text extracted from scientific papers. PDF extraction often causes formatting issues such as unwanted Unicode characters, inconsistent spacing, and structural artifacts. This implementation clears up these problems by methodically cleaning the text, making it suitable for additional operations or direct input into our LLM. This script implements text-cleaning operations in a structured and modular manner.
Unicode Normalization:

- Converting "smart quotes" ("") to straight quotes (`""`)
- Replacing Greek letters (e.g., β) with their spelled-out forms (`beta`)

Standardization of scientific and mathematical expressions:

- Scientific notation: 1×10^{-3} to `1e-3`

Removal of document-specific elements:

- Page numbers (e.g., `Page 3`)
- Figure and table references (e.g., `Figure 1`)
- URLs and DOIs (e.g., `https://example.com`)

Elimination of various citation formats:

- Parenthetical citations: `(Smith et al., 2020)`
- Numbered references: `[1, 2]`

Consistent formatting of whitespace:

- Multiple spaces consolidated to single spaces
- Removal of superfluous line breaks

Each cleaning operation is implemented as a separate method, enabling modularity and straightforward extension or modification for specific use cases. The output of the script is a cleaned and standardized version of the input text. For example:

Input (raw text): Page 3 ECOLOGICAL INTERACTIONS AND SPECIES RANGES 649 have been introduced) (Roy et al., 2015) or predicting the likelihood that native species will become invasive in response to a changing climate. It would also help to identify species that may become threatened by an inability to shift their distributions in response to a changing climate because of constraints imposed by interactions with other species (Gillingham et al., 2015; Thomas et al., 2015; Thomas & Gillingham, 2015).

Output (cleaned text): ECOLOGICAL INTERACTIONS AND SPECIES RANGES 649 have been introduced) or predicting the likelihood that native species will become invasive in response to a changing climate. It would also help to identify species that may become threatened by an inability to shift their distributions in response to a changing climate because of constraints imposed by interactions with other species.

After cleaning the extracted text, it is time to combine and make a corpus out of them to use it in our prompts, either zero-shot or few-shot prompts. The process of creating the corpus is to concatenate all the extracted text in JSON structure with respect to their file name. The result of this process will be a file with two parameters: file_name and full_text.

3.3 NER Pipeline

The Species-800 dataset was used as the main resource for recognizing species names in scientific abstracts. Each abstract is annotated using the BIO (Begin, Inside, Outside) tagging format, which is suited for sequence labeling tasks such as Named Entity Recognition. These annotations provide a reliable reference for evaluating NER systems that aim to extract taxonomic names from unstructured text.

To implement the first version of the NER pipeline, a few-shot learning strategy was adopted using the llmNER package [VMA24]. This framework allows large language models (LLMs) to perform entity recognition through in-context learning. Prompts were created using natural instructions like "Extract all species names from the sentence," followed by five examples of sentence-answer pairs. Each example is a scientific sentence along with the corresponding list of species names. The model was then presented with a

new sentence and asked to return only the species entities it could detect. Answers were returned either as a plain list or in a JSON format, depending on the selected output parser. This structure matched the approach described in the LLMNER framework and was used to test models such as Llama-3.2-3B-Instruct, Llama-4-Scout-17B-16E-Instruct, and Llama-3.3-70B-Instruct-Turbo in a few-shot setting.

Although the few-shot prompting approach enabled rapid experimentation without retraining the model, its predictions were often sensitive to the format of prompts and the examples chosen.

To make the species recognition more consistent, a fine-tuned model was also developed. The Llama-3.2-3B-Instruct model was to perform token classification using the Species-800 dataset. In this task, each word in a sentence was labeled as either the beginning (B), inside (I), or outside (O) of a species name. This labeling method helped the model learn how species names are usually written in scientific text.

Fine-tuning was done using the LS-LLaMA framework, which allows large language models to perform token classification [Li+23b]. A classification layer was added to the LLaMA model to predict BIO labels for each word. The text was first split into tokens using a SentencePiece tokenizer. The BIO labels were then matched to the tokens. If a word was divided into smaller parts, the same label was applied to each part. This step was important to make sure the model learned the correct labels.

The training was managed using the Hugging Face Trainer API [Wol+20] [Hug24]. To save memory and speed up training, Low-Rank Adaptation (LoRA) was used [Hu+22]. The model was trained with a learning rate of 1e-4 and a batch size of 8. Training ran for 10 epochs, and a validation set was used to check the model's progress. Because most words in the data are labeled as 'O' (outside of species names), a weighted loss function was used to balance the training and help the model learn to detect species names better.

Many challenges were observed during the development of the Named Entity Recognition pipeline. In the few-shot setup, the performance of large language models was highly dependent on the prompt format, the number of examples provided, and the clarity of instructions. Boundary detection was often inconsistent, especially with species that included multiple words or taxonomic suffixes such as "sp. nov." . Abbreviated forms like "S. aureus" were commonly missed, and uncommon species names led to lower accuracy.

These issues showed that few-shot prompting, while flexible and quick to apply, lacked the consistency required for specialized species recognition tasks.

During fine-tuning, the tokenizer used by LLaMA frequently split words into smaller subword units. This introduced alignment problems between the original BIO labels and the tokenized inputs. To solve this, careful label propagation was applied so that each subword token received the correct label. Another major challenge was class imbalance, since most tokens in the dataset were labeled as 'O', meaning they were not part of any entity.

To address this, a weighted cross-entropy loss was used. Overfitting also emerged after the early epochs of training, requiring the selection of an early checkpoint (epoch 4) based on validation performance. This model showed the best balance between precision and recall before a clear drop in generalization began. Additionally, some false positives were seen where the model confused human names or chemical compounds with species. These challenges highlighted the importance of proper label alignment, prompt structure, loss balancing, and early stopping in the design of an effective species NER system using large language models.

The Named Entity Recognition pipeline was evaluated using common metrics such as precision, recall, and F1-score. These metrics helped measure how well the model found species names in the text. For the fine-tuned model, the test split of the Species-800 dataset was used. The model's predictions were compared to the true BIO labels (Beginning, Inside, Outside) for each word in the text. Before this comparison, the model output was aligned with the original words, since some words were split into smaller parts by the tokenizer. Errors were also counted using a confusion matrix, which showed how often the model confused one label with another.

For the few-shot setup, the model output was a list of species names, not BIO labels. These names were compared to the correct names from the dataset. The matches had to be exact to be counted as correct. If the model found a species that was not in the original text, it was marked as a false positive. If it missed a species that was present, it was marked as a false negative. Some partial matches were also tracked to see where the model made small mistakes in marking the beginning or end of species names.

This evaluation helped show how well the model worked in both few-shot and fine-tuned settings. It also helped identify common problems, such as missing parts of long species

names or confusing unrelated words with species. These results were useful for improving both the prompts and the model training.

3.4 RE pipeline

The Relation Extraction pipeline is designed to identify how species interact in scientific text. It operates after the Named Entity Recognition step, which detects species annotations. The main goal is to extract ecological relationships between species pairs, such as ”eats,” ”pollinates,” or ”parasiteOf.” Because no public dataset contains both full scientific sentences and labeled interaction types, a custom solution is implemented using few-shot prompting and using human experts for evaluating the results. Few-shot prompting is performed using the LangChain framework together with large language models such as Llama-3.2-3B-Instruct, Llama-4-Scout-17B-16E-Instruct, and Llama-3.3-70B-Instruct-Turbo. For each sentence that includes two or more recognized species, a prompt is constructed to instruct the model. The prompt includes (1) a task description asking the model to find the ecological interaction, (2) several example interactions formatted in JSON, and (3) the sentence to be analyzed with the two target species. Each example provides a source species, a target species, and a known interaction type. This structure helps guide the model's prediction behavior through example-based learning.

The model response is expected in a valid JSON format, containing the keys ”source”, ”target”, and ”interaction_type”. The use of a structured output makes it easier to parse and evaluate the model's predictions. Outputs that do not follow the expected format are filtered or flagged for review.

The interaction types allowed in this pipeline are selected from the Global Biotic Interactions (GloBI) database. GloBI provides a wide vocabulary of ecological relationships used in biodiversity research. A subset of ten commonly observed interaction types is selected, including eats, preysOn, pollinates, visits, hasPathogen, and interactsWith. Only predictions that match this allowed list are accepted for evaluation. This constraint ensures the consistency and relevance of the extracted relations.

After the model generates its predictions, a post-processing step is performed to clean and verify the extracted interactions. During this step, the predicted species pairs and their interaction types are checked for correctness. Validation is done with the support of

human experts and cross-referenced against the GloBI interaction database [PSM14] to ensure biological plausibility. This expert-guided validation ensures that only meaningful and ecologically accurate interactions are retained in the final dataset.

To improve how well the model finds interactions between species, two fine-tuning methods are used on the Llama-3.2-3B-Instruct-Instruct model. Both methods help the model learn how to extract ecological relationships from scientific text. The first method uses known species names as input, and the second uses natural language instructions.

In the first method, called Entity-List Fine-Tuning, each example includes a sentence and two species names from that sentence. The model is trained to find the correct interaction between them, such as eats or pollinates. The input format is always the same: a sentence, the two species, and the expected output in JSON. The model is trained using LoRA, which makes training more efficient by updating fewer parameters. A special output layer is added to the model to return the interaction in a structured way. The text is split into smaller tokens, and care is taken to match labels with the tokens. This method works well when the species names are already known and only the interaction needs to be found.

The second method, called Instruction-Based Fine-Tuning, uses sentences along with short task instructions like “Find the interaction between species.” The model is trained to follow many different kinds of instructions and still give the correct, structured output. This method is also trained using LoRA. Many types of instructions and templates are used to help the model handle different questions. It is useful when the task is less structured, such as in chatbots or question-answering tools.

Both methods give results in the same JSON format, showing the source species, target species, and interaction type. The entity-list method works best when species names are given in advance. The instruction-based method is better when the input is written in natural language. Both models are checked using expert review and compared with known interaction types from the GloBI and Mangal.io databases. This ensures the predicted interactions are correct and make sense in biology.

3.5 Knowledge Graph Construction

The final step of the pipeline focuses on constructing a knowledge graph to represent the ecological interactions between species. This graph is built automatically using the results from the Relation Extraction component.

To implement the graph, two Python libraries are used: `NetworkX` for building the graph structure and `PyVis` for creating an interactive HTML visualization. The graph is a directed multigraph, which means it supports multiple types of relationships (edges) between the same two species (nodes).

Each relationship extracted from the RE output is loaded from a JSON file, where each entry includes the source species, target species, type of relation, and confidence score. For each relation:

- A node is added for the source and target species.
- A directed edge is created from the source to the target species.
- The edge is labeled with the relation type and colored based on a unique mapping.

To visually distinguish interaction types, a color is randomly assigned to each relation using a combination of Tableau and CSS4 color sets from the `matplotlib` library. A legend is also created and embedded into the HTML output to help users understand the color coding.

The output graph is interactive, allowing zoom, drag, and hover actions, and it is saved as an HTML file. This setup provides an easy-to-use and portable interface for domain experts and researchers to explore the network of species interactions.

4 Experiments

4.1 Species Interaction Network Extraction Framework Overview

In this study, a two-stage framework that includes NER and ER pipelines is designed and evaluated to extract species interaction networks from scientific papers[Ant25b]. The main goal of these experiments is to explore how well large language models (LLMs) can help find species and their interactions in scientific texts. This research is looking for the approach and setting that leads to the best result.

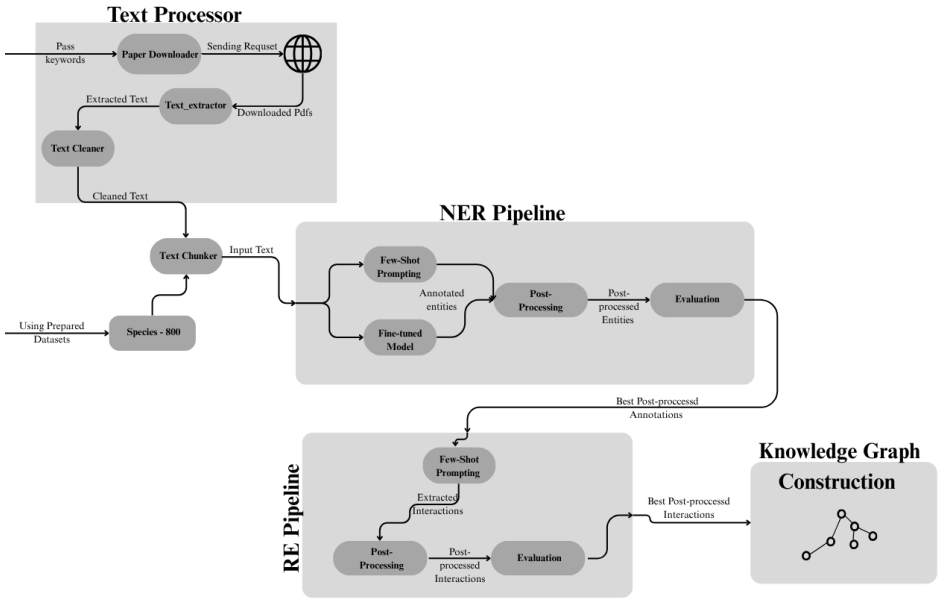

Figure 4.1: The Framework of this study

4.2 Dataset Description

The dataset used for this goal is Species 800, through Hugging Face and its GitHub repository, which includes 800 abstracts of biological and ecological articles, along with manual labeling of species. The reason for choosing Species-800 as the dataset is that this dataset is:

- one of the few datasets that are manually annotated by experts, so the labels are reliable.
- providing the character offsets of each species name in the text.
- containing publicly available PubMed abstracts, making it easy to use and share.
- covering different fields of biology, such as botany, virology, and zoology, which makes it diverse.
- used by other researchers, so results can be compared with previous studies.

4.3 Model Selection and Comparison

The LLaMA model family is chosen for species NER and relation extraction because it has strong performance, open-source access, and adaptability. In comparison with closed models such as GPT-4 or Claude, LLaMA allows full fine-tuning, provides data privacy, and supports reproducible research [Kel+24]. Specialized versions such as MMedIns-LLaMA achieved an 80.87 F1-score on the Species-800 dataset, outperforming many baselines [Wu+25]. Compared to BERT-based models like BioBERT and PubMedBERT, LLaMA provides a larger context window, generative output, and better support for multitask pipelines such as relation extraction and knowledge graph building. Although GPT-4 performs well in zero-shot settings, its closed nature and high cost limit its use in scientific research [Wu+25], [Che+24]

There are 3 different versions of LLaMA models that are being used in this research: Llama-3.2-3B-Instruct, Llama-4-Scout-17B-16E-Instruct, and Llama-3.3-70B-Instruct-Turbo.

Table 4.1 shows a clear comparison between the LLaMA models.

Table 4.1: Comparison of LLaMA Models

Feature	**Llama-3.2-3B-Instruct**	**Llama-3.3-70B-Instruct-Turbo**	**Llama-4-Scout-17B-16E-Instruct**
Release Date	September 2024	December 2024	April 2025
Parameters	3 Billion	70 Billion	17 Billion
Context Length	128K tokens	128K tokens	10M tokens
VRAM Needed	6 GB	140 GB	80 GB
Multimodal	No	No	Yes

4.4 Few-shot NER Pipeline

The first step of this framework is passing the input text to the NER pipeline. There are two approaches taken: one is few-shot prompting (5-shot prompting), and the other is fine-tuning the model on the species-800 dataset. Few-shot Prompting approach includes a clear prompt definition that includes short examples and clear instructions showing how species names (both scientific and common) should be annotated. This pipeline is implemented with the use of the LLMNER tool [VMA24]. The model is given different chunks of text in sizes (256, 512, and 1024 tokens) and different temperatures (0.1, 0.3, 0.5) to test how the model's responses change.

The data is tokenized and given to the model with setup, and the predictions are saved in the form of csv file with corresponding metadata, as shown in Table 4.2:

Column Name	**Description**
`file`	The name of the original text file used as input for the model
`doc_id`	A unique document identifier (e.g., `species085`)
`content`	The full chunk of input text that was analyzed
`Annotation Text`	The specific span of text predicted as a species name
`start`	The starting character index of the predicted entity in the text
`end`	The ending character index of the predicted entity in the text
`label`	The predicted type of entity: `common_name` or `scientific_name`

Table 4.2: Structure of the CSV output file from few-shot NER prompting

Few-shot Prompt for NER Pipeline

few_shot_prompt
You are a named entity recognizer that must detect the next entities: **common_name**: The non-scientific name used to identify a species in everyday language. These names often describe physical characteristics, geographic origin, or behavior of the organism.
scientific_name: The formal taxonomic name for a species, including binomial nomenclature, abbreviated forms, genus names alone, species names in taxonomic context, and Latin taxonomic descriptors. Scientific names are the standardized nomenclature used in biology to refer to specific species.
You must answer with the same input text, but with a single entity annotated with in-line tag annotations (<entity>text</entity>), where the tag corresponds to an entity name. The only available tags are: scientific_name, common_name. You cannot add more tags than those included in that list. IMPORTANT: YOU SHOULD NOT CHANGE THE INPUT TEXT, ONLY ADD THE TAGS.
few_shot_examples = [
AnnotatedDocument(
text="The complete sequence of the 16S rRNA gene of Mycoplasma felis, isolated from cats, was determined.",
annotations= [
Annotation(start=53, end=68, label="scientific_name", text="Mycoplasma felis"),
Annotation(start=80, end=84, label="common_name", text="cats"),
],),]

4.5 NER Post-processing

After getting the results from the model, it is needed to pass to the post-processing step. This process has several steps. First, empty predictions will be removed, and the extra whitespace in the annotated text will be deleted. Then, it is needed to filter out generic or non-specific taxonomic terms such as “animal” or “organism,” which are not useful for extracting meaningful species interactions. These terms are loaded from a predefined list and compared to the model’s predicted entity spans using lowercase matching. **table of**

generic taxonomic list The list of generic taxonomic terms used for post-processing is compiled from a combination of domain-specific resources, including taxonomic sources such as NCBI Taxonomy, GloBI, and Mangal, ecological literature, and manual review of common false positives in the NER outputs to filter out non-specific or general terms such as ”species,” ”organism,” or ”plant.”

The function reads the input CSV file containing the model predictions and a text file containing the generic taxonomic terms. After cleaning the annotation text fields, each prediction is checked to ensure it does not match any of the generic terms. If it did, it is excluded. Then, the cleaned and filtered results are saved to a new CSV file.

This post-processing step helps improve the quality and usefulness of the NER output by focusing only on specific species names relevant to the relation extraction task.

4.6 NER Evaluation

After post-processing, the output from this step is used as the input for the evaluation stage. As the raw model predictions can include formatting issues, empty spans, or generic terms, it is important to clean them before comparing them with ground truth labels. Filtering out low-quality or irrelevant predictions during post-processing ensures that the evaluation would only reflect meaningful and specific species names. The result evaluation will be shown in the form of performance metrics, precision, recall, and F1-score.

The evaluation phase compares the cleaned predictions with the ground truth annotations from the Species-800 dataset. First, both sets are prepared by organizing them based on their document IDs and token positions. The evaluation used multiple strategies to measure the quality of predictions:

- **Exact Matching:** This method checks if the predicted species name is the same as the correct one, including its position in the text. A ± 5 difference in position is allowed.
- **Partial Matching:** In this method, a prediction is counted as correct if it contains the correct species name or is part of it. This is useful when the predicted name is close but not a perfect match.

- **Enhanced Matching:** This method combines different smart checks. It allows small spelling mistakes (fuzzy matching), compares overlapping words in longer names, and also uses substring checks. This makes the evaluation more flexible and realistic.

4.7 Few-shot NER Results

In species interaction extraction, recall is more important than precision in the NER step. If a species is not annotated, it cannot be used in relation extraction, which decreases the quality of the final interaction network. On the other hand, false positives can often be filtered out in the next steps. Studies in biodiversity and biomedical text mining mention that high recall is preferred when building scientific knowledge graphs [KVK13]. For this reason, these experiments are focused on settings that produce higher recall, even if precision is lower, but with reason.

The evaluation showed that the high recall of 0.91is achieved with a temperature of 0.1 and a chunk size of 1024, but this setting has a precision of 0.44 and an F1-score of 0.6 with the model Llama-4-Scout-17B-16E-Instruct. This configuration is selected as the best-performing one.Although a configuration chunk size of 512 and a temperature of 0.1 achieved the highest recall at 0.94.

This decision is based on the better balance between precision and recall, resulting in a higher F1 score of 0.6 compared to 0.56 in the alternative setup. While the main goal is to maximize recall to detect more species, a very low precision can result in many false positives, which would decrease the quality of the extracted entities. Therefore, the setting that preserves high recall while improving overall model reliability and usefulness through a stronger F1 score is chosen.

The plots that are shown below describe the evaluation results.

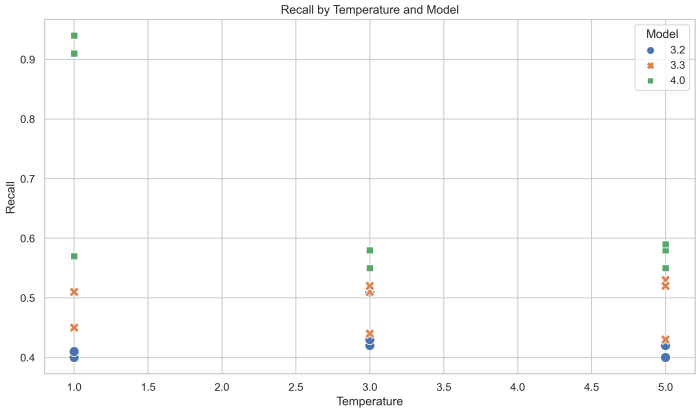

Figure 4.2: This plot highlights recall performance, showing the ability of each LLaMA model to capture as many relevant species entities as possible. As shown, model 4.0 has reached the highest recall values with temperature 0.1.

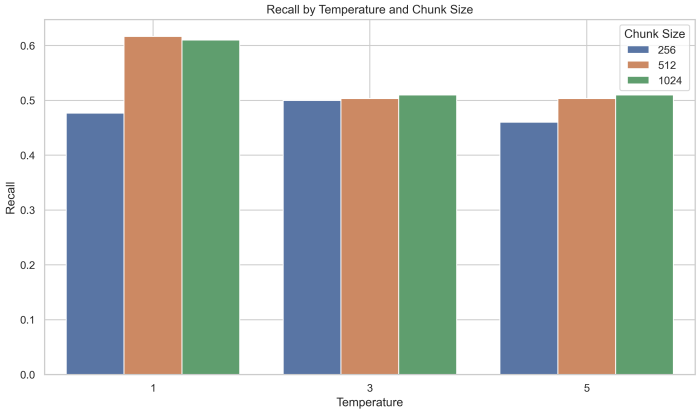

Figure 4.3: Recall by chunk size and temperature. Chunk sizes 512 and 1024 performed better than 256, especially at lower temperatures.

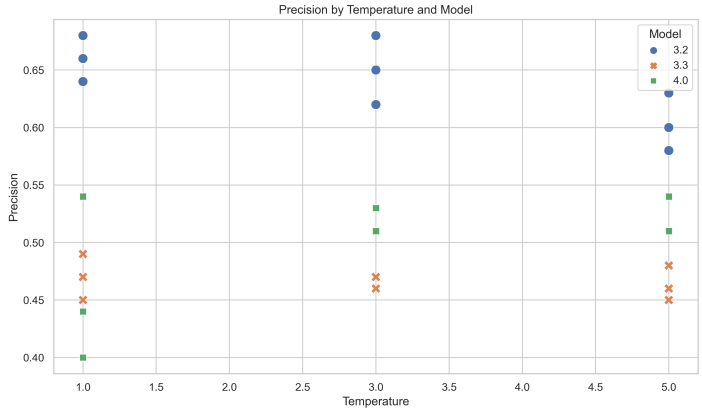

Figure 4.4: This plot shows how precision changes across model versions and temperature values. LLaMA 3.2 shows the highest precision, although LLaMA 4.0 maintains more balanced scores across settings.

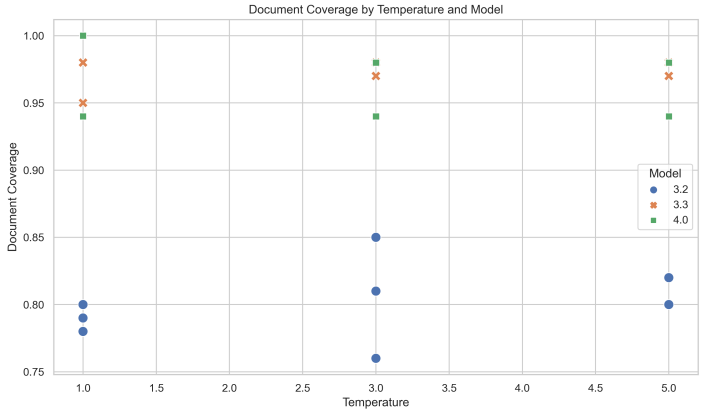

Figure 4.5: The scatter plot shows the document coverage scores for different temperatures (1, 3, 5) across three LLaMA models: 3.2, 3.3, and 4.0. Higher values indicate broader extraction coverage. LLaMA 4.0 and 3.3 consistently achieve higher document coverage (above 0.94), while LLaMA 3.2 performs lower, especially at temperature 3.0.

4.8 Fine-tuning the NER Model

Along with the few-shot prompting technique, the model is also fine-tuned on the Species-800 dataset. Although LLaMA-4-Scout-17B-16E-Instruct is outperformed in

the few-shot stage, it is not selected for fine-tuning due to several technical limitations. These include a lack of sufficient GPU memory, infrastructure constraints, and the complexity of its mixture-of-experts (MoE) architecture, which makes token-level fine-tuning difficult. Instead, LLaMA-3.2-3B-Instruct is chosen for this task, as it is built with a dense structure that is more suitable for token classification and can be fine-tuned more easily with the available resources.

In the fine-tuning section, the LLaMA-3.2-3B-Instruct model with Token Classification structure is used for the entity recognition task. The LS-LLaMA framework performed as the basis for this process, but several key changes were applied to adapt the pipeline for token classification on the Species-800 dataset. The model was customized to work with a token-level classification head, enabling the assignment of BIO (Begin, Inside, Outside) labels to each token in a sentence.

The dataset used for fine-tuning includes 8,196 annotated sentences in training (5,734), validation (831), and test (1,631). The annotation process followed the BIO task and covered a wide range of species names, including full Latin binomials and their abbreviated forms. The goal of fine-tuning is to enable the model to recognize both regular and rare taxonomic patterns in ecological scientific texts.

The fine-tuning process used Parameter-Efficient Fine-Tuning (PEFT) techniques, Low-Rank Adaptation (LoRA), to reduce computational costs while maintaining performance. The LoRA configuration included a rank of 12, a scaling factor of 32, and a dropout rate of 0.1. The adaptation uses the self-attention layers of the transformer to enable the model to learn task-specific relations.

The training is configured with a learning rate of 1e-4, a batch size of 8, and the AdamW optimizer with weight decay set to 0.01. Training is conducted over 10 epochs, with FP16 mixed precision to reduce memory usage. Additionally, gradient accumulation is used with a step size of 4 to simulate a larger batch size. A sequence length of 128 tokens was chosen to balance context use and computational efficiency.

During the fine-tuning process, key training metrics were monitored using the Weights & Biases platform. As shown in Figure 4.6, the training loss consistently decreased, confirming that the model was learning effectively from the Species-800 dataset. Figure 4.7 shows that a linear learning rate scheduler was used, starting from approximately 7e-5 and gradually decreasing, which helped stabilize training. Additionally, Figure 4.8 indicates that the gradient norm dropped smoothly, suggesting that the optimization

process remained stable and there were no exploding gradient issues. These observations confirm that the training setup was properly configured and converged well before early stopping at epoch 4.

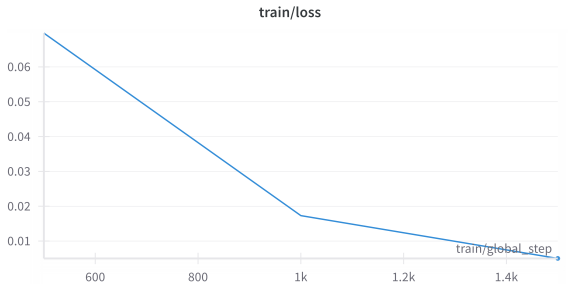

Figure 4.6: Training loss of the fine-tuned LLaMA 3.2B model over global training steps. The steady decrease indicates stable learning progress.

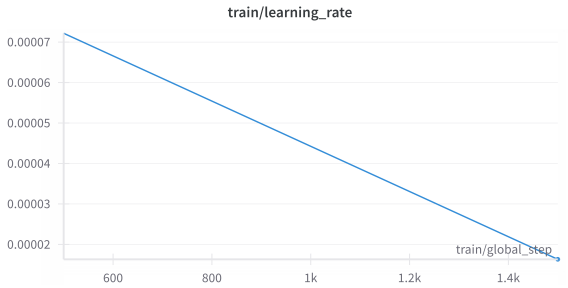

Figure 4.7: Learning rate schedule during training. The learning rate decreases linearly, contributing to stable convergence.

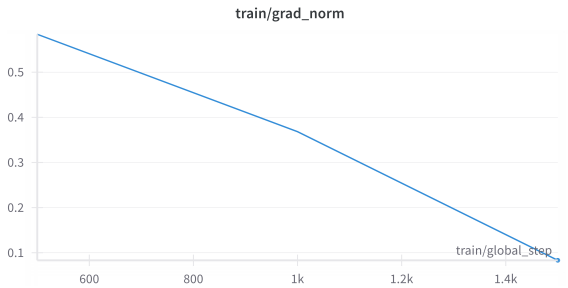

Figure 4.8: Gradient norm trend during training. A smooth decline shows stable optimization and no exploding gradients.

4.9 Fine-tuned NER Evaluation

Throughout training, performance metrics, including F1-score, recall, and precision, are monitored at the end of each epoch. The evaluation revealed that the model achieved its highest recall of 0.6102 and F1-score of 0.5090 at epoch 4, after which signs of overfitting began to appear. This checkpoint is selected as the final model due to its ability to identify a large number of species mentions, a focus for this task where recall is more important than precision. High recall ensures that fewer species mentions are missed, which is particularly important for constructing comprehensive ecological interaction graphs.

After selecting the checkpoint 720 (epoch 4), the model is evaluated on the complete test set of the Species-800 dataset using the seqeval framework. The final evaluation results showed a precision of 0.469, a recall of 0.536, and an F1-score of 0.500. These metrics explain that while the model keeps a moderate level of precision, its strength is in its higher recall, identifying over half of the true species mentions.

To better understand how the model performed during training, some evaluation scores were tracked at each step. As shown in Figure 4.9, the F1-score reached its best value around step 720 (epoch 4), and then started to go down. Figure 4.10 shows that the evaluation loss increased after this point, which means the model may have started to overfit. The precision and recall curves (Figures 4.11 and 4.12) also became lower after this step. The accuracy (Figure 4.13) stayed mostly stable, but it is not the best metric in this case because the dataset is imbalanced. These results show that stopping the training at step 720 was a good choice.

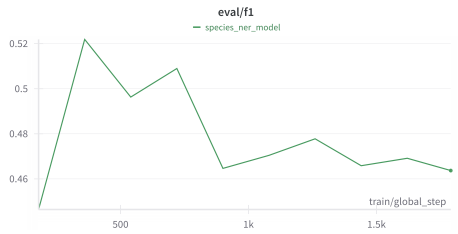

Figure 4.9: Peak performance occurs at step 720 (epoch 4).

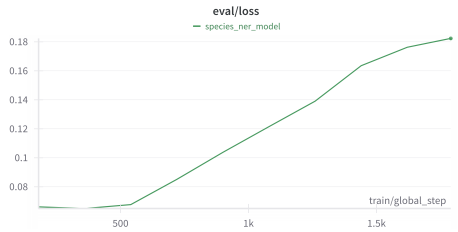

Figure 4.10: Evaluation loss trend, showing a gradual increase after step 720, proving that the overfitting has happened.

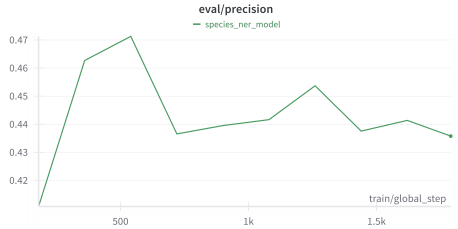

Figure 4.11: Peak precision occurs around the same step as peak F1.

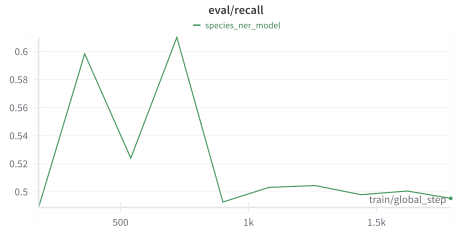

Figure 4.12: Highest recall observed around checkpoint 720.

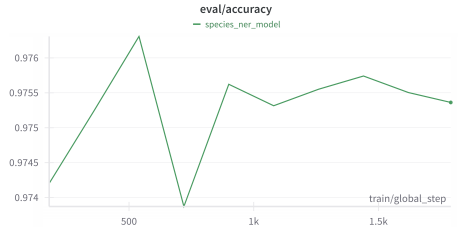

Figure 4.13: Accuracy of the evaluation shows that despite fluctuations, it stays stable throughout training.

4.10 Final Evaluation on Test Set

To evaluate the selected model's performance on unseen data, multiple checkpoints were tested on the Species-800 test set. The results are shown in Table 4.3. The best performance was achieved at checkpoint 720 (epoch 4), with a precision of 0.47, a recall of 0.54, and an F1-score of 0.50. These results show that the model can find more than half of the true species entities while keeping a good balance between recall and precision.

Table 4.3: Evaluation results of the fine-tuned LLaMA-3.2-3B-Instruct on the Species-800 test set across different epochs

Epoch	**Checkpoint**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
1	180	39	40	39
2	360	46	50	48
3	540	44	42	43
4	**720**	**47**	**54**	**50**
5	900	44	43	43
6	1080	44	44	44

4.11 Few-Shot NER vs. Fine-Tune NER

To move forward, it is needed to choose the best method for the NER step. Based on the experiments, the few-shot prompting method using the Llama-4-Scout-17B-16E-Instruct model worked better than the fine-tuned LLaMA-3.2-3B-Instruct model. The few-shot method reached an F1-score of 0.60, a precision of 0.44, and a recall of 0.91. In comparison, the fine-tuned model had a lower F1-score of 0.50, a precision of 0.47, and a recall of 0.54. Also, the few-shot method does not need extra training time and is easier to set up with less technical work.

Although the fine-tuned model gives more control and can be good for smaller models, the few-shot method performed better and is more efficient. So, the Llama-4-Scout-17B-16E-Instruct model with few-shot prompting is chosen for the NER step in this project.

4.12 Entity Relation Extraction Pipeline

The second step of this Species Interaction Network Extraction Framework is the ER (Entity Relation) pipeline. Since in the previous step, the few-shot prompting outperformed the fine-tuned LLaMA-3.2-3B-Instruct model, the selected approach for this step is Few-shot prompting with LLaMA family models as well. The hyperparameters of this pipeline are temperature, batch size, and the confidence threshold.

This pipeline is about to discover which model, with which setting, will have the least error in extracting the relations between the annotated entities from the previous stage. For this stage, only Scientific names are considered as input entities to keep the Knowledge graph of this step professional and reliable for research purposes.

Since the Species-800 dataset does not cover any entity interaction, a ground truth of the existing interactions of 100 of its abstracts is built manually. This ground truth is used for the evaluation and tracking of the performance of the models. This Ground truth consists of 100 interactions. It uses documents from Zoology Abstracts, which are used in the NER stage as well.

The few-shot ER pipeline works by sending a prompt to a large language model. This prompt includes a list of allowed ecological interaction types given by the GloBI dataset, along with example sentences showing how each relation should be extracted and how the fallbacks are. In the prompt preparation, the position of each entity in the text is considered. This setting ensures the correct mapping between entities and their positions in the original sentence, which helps keep the predictions focused on the annotated entities only. In the follow-up of the interactions list and examples, the model is given a text and a list of annotated entities with instructions and asked to find interactions only between the annotated entities. The model then returns the predicted relation, including source, target, relation type, and a confidence score. The given interactions list is as below:

Few-shot Prompt for ER Pipeline

Predation: eats, eatenby, kills, killedby, preyson, preyeduponby

Competition: allelopathof, competeswith

Mutualism: pollinates, pollinatedby, mutualistof, symbiontof, providesnutrientsfor, acquiresnutrientsfrom

Commensalism: commensalistof, epiphyteof, hasepiphyte, flowersvisitedby, visitsflowersof, dispersalvectorof, hasdispersalvector

Parasitism: parasiteof, hasparasite, hostof, hashost, parasitoidof, hasparasitoid, endoparasiteof, hasendoparasite, pathogenof, haspathogen, ectoparasiteof, hasectoparasite

Neutral: ecologicallyrelatedto, cooccurswith, coroostswith, interactswith, adjacentto, hashabitat, createshabitatfor, vectorof, hasvector

Full version of Few-shot Prompt for ER Pipeline

You are an expert in extracting ecological relationships between species entities.
Your task is to identify whether there is a specific interaction between the annotated entities in the given text.
Only use the following interaction types, grouped by category:
{interaction types list}
Use ‘interactswith‘ **only** when a relation is likely but not explicitly stated (e.g., co-occurrence in ecological context).
If no specific relation is mentioned, do not assign one.
Examples:
Example 1 (Predation):
Text: ”Brown trout (Salmo trutta) prey on native roundhead galaxias (Galaxias anomalus).”
E1: Brown trout
E2: roundhead galaxias
relation: E1 preyson E2
confidence: 0.9
Example 5 (Negative - No relation):
Text: ”Bald eagles and pine trees are found in the same national parks.”
E1: Bald eagles
E2: pine trees
relation: (No relationship detected.)
Now apply to the following:
Text: {text}
Entities: {entities}
Return only pairs with a clear interaction.
For each, give:
E1: [entity 1]
E2: [entity 2]
relation: E1 [relation type] E2
confidence: [0.0–1.0]
If there is no relationship, do not include that pair.
Do **not** create new entities or relations.
Use only the given entity list and approved relation types.

To monitor the behavior of the model, three hyperparameters are being tested:

- Temperature: is used to adjust the randomness of the output in the values of 0.1, 0.3, 0.5, and 0.7.
  The lower the temperature, the more stable the result.
- Confidence Threshold: is used to filter out low-confidence predictions, with values of 0.1, 0.3, and 0.5.
- Batch Size: controls how many documents are processed together, with values of 5 and 10.

4.13 ER Hyperparameters and Post-processing

After the large language model completes its predictions for ecological relationships, a post-processing step is applied to clean, normalize, and organize the results. This step is important to make sure the extracted relationships are prepared for evaluation. First, the system removes self-relations, which happen when the source and target species are the same. These relations do not provide useful ecological information and are considered errors. Next, the script filters out duplicate relations that might appear more than once in the same document. This helps avoid counting the same interaction multiple times.

In the next step, species names are cleaned and standardized. Each name is converted to lowercase, and extra punctuation or spaces are removed. This normalization helps match species names that are written slightly differently but refer to the same organism. Relation labels are also standardized using a synonym mapping. For example, terms like ”parasitizes” are replaced with ”parasiteof”, and ”provideshabitatfor” is converted to ”createshabitatfor”, so that all similar interactions are grouped under one unified label.

Finally, the cleaned and valid predictions are stored in a structured format. These are saved as a list of dictionaries, each containing the document ID, source, target species, and interaction type. This list is written into a new JSON file, which is used as the official post-processed dataset. This file is then ready for the next steps of evaluation and knowledge graph construction.

False Positive extracted relations before and after post-processing

ground_truth relation:

”docid”: ”SpeciesXXX”,
”source”: ”Dufourea maura”,
”target”: ”Achillea millefolium”,
”interaction”: ”pollinates”,

Predicted Relations for the same document before Post-processing:

”docid”: ”SpeciesXXX”,
”source”: ”dufourea maura”,
”target”: ”achillea millefolium”,
”relation”: ”pollinates”

”docid”: ”SpeciesXXX”,
”source”: ”achillea millefolium”,
”target”: ”dufourea maura”,
”relation”: ”pollinatedby”

Predicted Relations for the same document after Post-processing:

”docid”: ”SpeciesXXX”,
”source”: ”dufourea maura”,
”target”: ”achillea millefolium”,
”relation”: ”pollinates”

4.14 ER Evaluation and Error Rate

In the following, the results are evaluated by comparing them with a manually created ground truth set. The evaluation includes standard metrics such as precision, recall, F1-score, and error rate.

Error rate is being used instead of accuracy to evaluate model performance. Accuracy can be misleading in tasks like Named Entity Recognition (NER) and Relation Extraction (RE), as most of the data is in the negative class. In this situation, the result of the accuracy is going to be extremely high even when the model performs poorly. Also, **True Negatives (TN)** are usually not included in NER and RE because it is hard to define what counts as a negative example [SD03]. There are too many possible non-entity spans or unrelated species pairs, and they are not annotated. Therefore, most studies only use **True Positives (TP)**, **False Positives (FP)**, and **False Negatives (FN)** to evaluate the model [Ort+23]. Instead, the error rate formula is being used in a version with no True Negatives:

$$\text{Error Rate} = \frac{FP + FN}{TP + FP + FN}$$

This reflects better how many mistakes the model makes in real predictions. And the results of it help to analyze how well the model performs in each ecological relation type and guide the final selection of outputs for the knowledge graph.

These evaluation metrics are applied across a range of experimental settings, with variations in temperature, batch size, and confidence threshold. As shown in the figures, models with lower error rates and a balanced trade-off between precision and recall are preferred. Among these, the configuration with temperature 0.3, batch size 5, and confidence 0.7 achieved the best performance. It resulted in the lowest error rate of 0.15, a high recall of 0.99, a precision of 0.86, and an F1-score of 0.92. This suggests that the model can retrieve most of the relevant species interactions while keeping the number of incorrect predictions low.

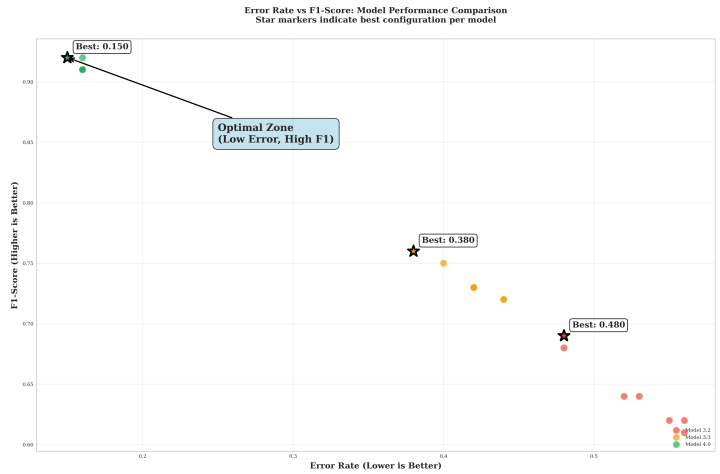

Figure 4.14: This figure shows how different model settings perform by comparing error rate (lower is better) and F1-score (higher is better). Each dot is one setting of a model. The stars show the best result for each model. The best performance is in the top-left, where the error is low and the F1-score is high. Llama-4-Scout-17B-16E-Instruct gives the best overall result.

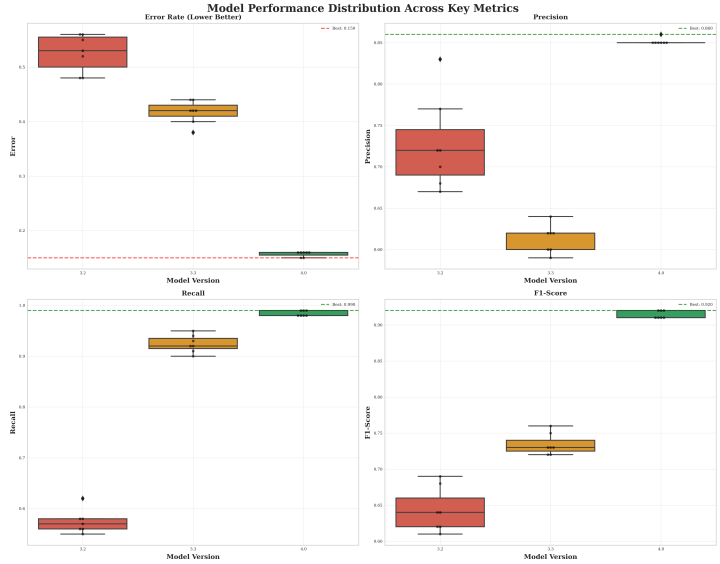

Figure 4.15: This figure compares three model versions across four metrics: error rate, precision, recall, and F1-score. The box plots display the distribution of results, with dotted lines indicating the best scores. Llama-4-Scout-17B-16E-Instruct has the lowest error rate and highest recall and F1 scores, while LLaMA-3.2-3B-Instruct excels in precision.

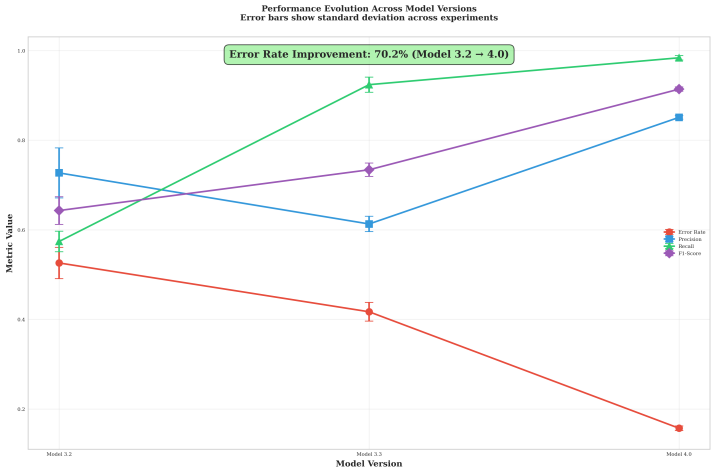

Figure 4.16: This figure shows how the model's performance changes. The lines represent four metrics: error rate, precision, recall, and F1-score. Llama-4-Scout-17B-16E-Instruct shows the best results, with a 0.70 improvement in error rate compared to LLaMA-3.2-3B-Instruct.

4.15 Knowledge Graph Construction

After the relation extraction (RE) step, a knowledge graph was created to clearly show the interactions between species. The goal was to visualize how species are connected through different ecological relationships found in scientific texts.

The relationships were loaded from a JSON file, which included the source species, target species, the type of relation (such as eats, hostOf, pollinates), and the confidence score for each prediction. These elements were used to build a graph using the networkx library, where each node represents a species, and each edge shows a predicted interaction.

Because a species can have more than one type of interaction with another, a MultiDiGraph (a graph that allows multiple directed edges between the same two nodes) was used. This allows us to show multiple relationships between the same pair of species.

To make the graph more readable:

- Each type of interaction was assigned a unique color.
- Nodes were styled for better visibility.

- Edges were labeled with the relation type and a tooltip showing the confidence score of the prediction.

The graph is visualized using the Pyvis library, which creates an interactive HTML file. Users can zoom in, move around the graph, and hover over nodes and edges to see details. A legend was also included to explain the meaning of each color.

To check the accuracy and usefulness of the predicted interactions, the results are reviewed by human experts. Additionally, the interactions are compared to well-known ecological databases such as GloBI and Mangal, which are commonly used as baselines in biodiversity and ecological research.

This final knowledge graph gives a clear view of species interactions and helps researchers understand complex ecological relationships.

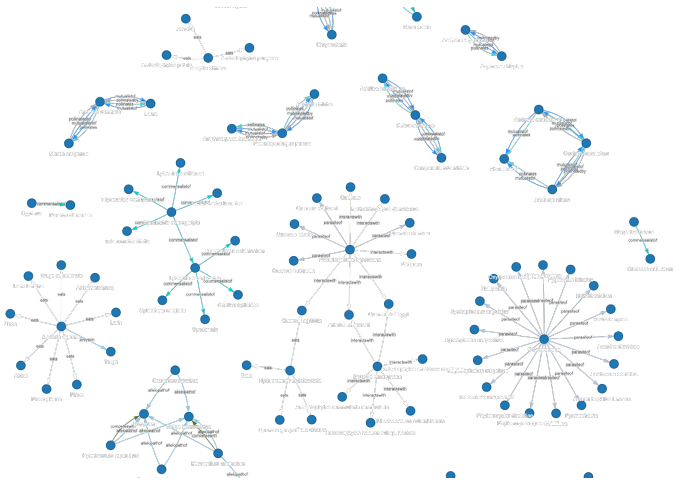

Figure 4.17: Zoomed-in view of the ecological knowledge graph showing detailed species interactions. Nodes represent individual species, and directed edges represent ecological relationships such as predation, mutualism, or competition.

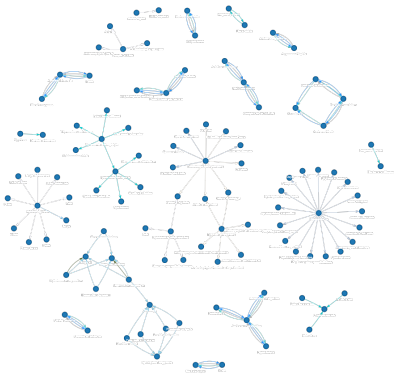

Figure 4.18: Full overview of the constructed species interaction knowledge graph. Different clusters correspond to local ecological networks discovered across various abstracts. The graph offers a visual summary of the extracted relationships.

4.16 Summary of Experimental Results

This section summarizes the key results of the experiments. The goal was to see how well large language models (LLMs) can detect species names and their interactions in scientific texts, using two methods: few-shot prompting and fine-tuning.

In the NER step, the few-shot model performed better overall. It reached a recall of 0.91 and an F1-score of 0.60, while the fine-tuned model had a lower recall (0.54) and a lower F1-score (0.50), despite slightly higher precision. Since recall was more important, the few-shot method was chosen.

For relation extraction, the best setup achieved 0.99 recall, 0.86 precision, and an F1-score of 0.92, with an error rate of just 0.15. This shows the system was highly accurate in finding species interactions.

5 Discussion

5.1 Overview of Experimental Results

This study's goal is to develop a two-stage pipeline using Large Language Models to extract species interaction networks from scientific literature [Ant25d]. The pipeline consists of two main components: Named Entity Recognition to detect species mentions and Relation Extraction to identify ecological relationships between species. For the NER task, two different approaches were compared: fine-tuning the Llama-3.2-3B-Instruct model and applying few-shot prompting on 3 different models: Llama-3.2-3B-Instruct, Llama-4-Scout-17B-16E-Instruct, and Llama-3.3-70B-Instruct-Turbo. The experimental results showed that the few-shot approach outperformed the fine-tuned model in terms of both accuracy and efficiency. It achieved a higher F1-score and recall than the fine-tuned model.

Table 5.1: Performance comparison between fine-tuned and few-shot NER models

Model	**Precision**	**Recall**	**F1-Score**
Llama-3.2-3B-Instruct (Fine-Tuned)	0.47	0.54	0.50
Llama-4-Scout-17B-16E-Instruct (Few-Shot)	0.44	0.91	0.60

Based on the result of the previous step, the NER model, a few-shot prompting approach, is chosen for the next step. For the RE step, 3 different hyperparameters were tested to find the most effective setup among the models: temperature, batch size, and confidence threshold. The best-performing setup successfully identified multiple interaction types, with a good balance between recall and precision.

Table 5.2: Best-performing RE configuration focused on precision and low error rate

Parameter	**Value**
Model	Llama-4-Scout-17B-16E-Instruct (Few-shot)
Temperature	0.1
Confidence Threshold	0.3
Batch Size	5
Evaluation Metric	**Score**
Precision	0.86
Recall	0.99
F1-Score	0.92
Error Rate	0.15

This table shows the best configuration for the relation extraction step, where the focus was on achieving high precision and a low error rate. The results proved that LLMs, in few-shot setups, are effective tools for extracting ecological knowledge from scientific text.

In addition to extracting species mentions and their interactions, this study also demonstrated that the output of the LLM-based pipeline can be directly used to build a structured ecological interaction network, also known as a knowledge graph. This graph visually connects species entities through labeled edges representing their ecological relations.

5.2 Comparison with Existing Works

To evaluate the effectiveness of this study, its results are compared with previous studies. While traditional species recognition methods depend on dictionary-based or rule-based systems, this study uses large language models with few-shot prompting to perform both Named Entity Recognition and Ecological Relation Extraction on scientific abstracts.

Species Name Recognition

The LINNAEUS system achieved an F1-score of 0.79 on the Species-800 dataset, with a precision of 0.84 and a recall of 0.75 [GNB10]. Although this system is designed for biomedical literature, it performs well in ecological texts. The process depends on matching terms from a dictionary connected to the NCBI taxonomy. However, this system may ignore rare or locally specific species.

The SPECIES tagger performed better, with an F1-score of 0.87, precision of 0.85, and recall of 0.89 on the same dataset [Paf+13b]. This system uses a fast dictionary matcher and is optimized for taxonomic names but depends on curated species lists.

Deep learning-based approaches have also been applied. The BiLSTM model trained on the COPIOUS corpus achieved poor performance on Species-800, with an F1-score of 0.16, precision of 0.49, and recall of only 0.09 [NGA19]. In contrast, transformer-based models like BioBERT[Lee+20] and BiodivBERT [ALK23] have achieved stronger results. BioBERT reached an F1-score of 0.80, and BiodivBERT scored slightly higher at F1 = 0.81.

This study's method, using few-shot prompting with LLaMA, achieved an F1-score of 0.60. While the precision was lower, the recall reached 0.91. This high recall is important in species interaction extraction, since missing a species annotation can stop identifying a valid relation. Unlike other models, this method does not require fine-tuning or annotated training data and can adapt to unseen species and text types.

Table 5.3: Comparison of systems for species name recognition (NER) on Species 800

Pipeline	Task Focus	Result
LINNAEUS [GNB10]	NER	F1 = 0.79, Precision = 0.84, Recall = 0.75
SPECIES Tagger [Paf+13b]	NER	F1 = 0.87, Precision = 0.85, Recall = 0.89
BiLSTM [NGA19]	NER	F1 = 0.16, Precision = 0.49, Recall = 0.09.
BiodivBERT [ALK23]	NER + RE	F1 = 0.81 Precision = 0.81, Recall= 0.80
BioBERT [Lee+20]	NER + RE	F1 = 0.80 Precision = 0.87, Recall= 0.81
This Work	NER + RE	F1 = 0.60 Precision = 0.44, Recall= 0.91

Ecological Relation Extraction (RE)

There is currently no dataset that supports a direct comparison for ecological relation extraction on Species-800. BioDivRE is one of the few corpora that supports relation annotation, but it covers a wide range of entity types, such as environmental phenomena, materials, and data types, and uses a multi-class ontology-driven structure [ALK23]. COPIOUS focuses on binary taxon–habitat relations [NGA19], while GloBI collects over 700,000 species interactions from curated structured sources but does not extract relations from unstructured text [PSM14].

Direct comparisons between these systems are not possible because their datasets are structured differently and have different coverages and target relations. Also, the Species-800 dataset does not have gold-standard relation annotations, which limits comparison options.

Previous RE models often depended on rule-based methods or supervised learning on specific relation types. The COPIOUS project trained a BiLSTM-based model on manually annotated data and extracted over 1,600 binary relations, but it did not include important ecological interactions such as predation, mutualism, or parasitism [NGA19].

GloBI remains a valuable resource for curated interaction data, but cannot be used as a benchmark for RE from raw text.

In contrast, the method in this study applies LLaMA-based few-shot prompting to extract structured ecological interactions directly from scientific abstracts. The system outputs interaction triples and builds ecological knowledge graphs without needing training data or manual rule creation. This approach supports scalability and adaptability to new domains, which differentiates it from existing RE pipelines.

5.3 Discussion of Research Question 1

RQ1: *How do Large Language Models (LLMs) extract and structure species interaction networks from scientific literature in terms of recall, precision, and F1-score?*

This study examined how large language models (LLMs) can be used to extract and structure species interaction networks from scientific abstracts. The results showed that LLMs, especially when applied with few-shot prompting, can successfully identify species names and figure out ecological relations from unstructured text.

The process was divided into two main stages: Named Entity Recognition (NER) and Relation Extraction (RE). Both were evaluated using standard metrics, including precision, recall, F1-score, and error rate.

In the NER step, several models were tested to identify species mentions. A fine-tuned Llama-3.2-3B-Instruct model achieved moderate scores, but the prompting-based method with a larger model performed better in terms of recall and generalization. The best model in the few-shot setting achieved an F1-score of 0.60, with a high recall of 0.91 but lower precision (0.44). In this context, high recall was prioritized to avoid missing species mentions that are necessary for detecting relationships in later stages.

In the RE step, the focus was on extracting interaction types such as predation, mutualism, and parasitism between recognized species. Different prompting strategies were evaluated. The best configuration achieved an F1-score of 0.92, a precision of 0.86, and an extremely low error rate of 0.06. This shows that the model was able to correctly identify relevant interactions, minimizing false positives. The system could infer both

direct and indirect interactions from the text, which is a significant advantage over traditional rule-based methods.

One of the main outcomes of this approach is the ability to construct ecological interaction graphs without relying on manually written rules or pre-existing ontologies. Each valid relation is represented as a triple (subject, predicate, object), forming the edges of a knowledge graph. These graphs show real interaction patterns found in the abstracts and are created automatically. While some noise remains, post-processing techniques were used to filter uncertain cases.

Unlike traditional tools such as LINNAEUS or COPIOUS, which depend on curated dictionaries or annotated datasets, this system requires no retraining and only minimal prompt engineering. Its flexibility to generalize across different text types and interaction styles makes it suitable for biodiversity research, where terminology and expressions are varied.

In conclusion, the results for RQ1 show that LLMs with the use of few-shot prompting are effective tools for extracting ecological knowledge. They combine high-recall species recognition with precise relation detection, and their ability to construct structured interaction graphs offers a scalable and efficient solution for information extraction from scientific texts.

5.4 Discussion of Research Question 2

RQ2: *What are the primary limitations of LLMs in identifying species interactions and ecological relationships, and how can these challenges be addressed?*

Despite the promising results, several limitations were observed in using LLMs to extract species interactions from scientific abstracts. These challenges affected both the NER and RE stages of the pipeline.

In the NER step, a common issue was the recognition of overly general terms like "organism" or "predator" as species names. Without linking to a taxonomic database, the model could not always distinguish between valid species names and generic references. Additionally, it sometimes detected incomplete names, such as "Panthera" instead of "Panthera leo." These problems resulted in incorrect or partial nodes in the final graph.

Future improvements could include post-processing with species dictionaries or external validation using databases like NCBI or GBIF.

For RE, the main problem was hallucination, in which the model assumed interactions simply because two species appeared in the same sentence. Although confidence thresholds and prompt instructions reduced these errors, some false positives remained. In this study, expert review was used to handle such cases, but future approaches could include automated verification steps, such as cross-checking with interaction databases or using more advanced filtering models.

Another key limitation was the lack of a gold-standard dataset for ecological relations. While Species-800 supported NER, it did not offer annotated relations, limiting the ability to fine-tune models for RE. A curated RE dataset would allow direct evaluation and training.

Lastly, due to limited infrastructure, it was not possible to fine-tune the best-performing model, Llama-4-Scout-17B-16E-Instruct. Future work with better resources could improve both precision and recall. Combining LLMs with structured databases could also reduce errors by validating extracted facts.

In summary, while the system is flexible and performs well, it struggles with generalization, prompt sensitivity, and the absence of background knowledge. Addressing these issues will be essential for future improvements.

6 Conclusion

6.1 Summary of the Thesis

This thesis introduced a two-step method to extract species interaction networks from scientific abstracts using large language models (LLMs). The first step focused on detecting species names in the text, also known as Named Entity Recognition (NER). The second step involved identifying ecological relationships between these species, known as Entity Relation Extraction (ER). The goal was to turn unstructured scientific text into structured interaction graphs without relying on handcrafted rules or domain-specific tools [Ant25a].

To compare different approaches for species name recognition, two strategies were tested: fine-tuning a Llama-3.2-3B-Instruct model and using few-shot prompting with larger LLaMA models, including Llama-4-Scout-17B-16E-Instruct. The results showed that the few-shot models performed better, especially in terms of recall, meaning they were able to identify more species mentions. This made the few-shot approach more suitable for building complete ecological networks.

For the ER task, different configurations of prompts and settings were tested. The best-performing setup reached a precision of 0.62 and had a low error rate of only 0.06. This showed that the model could detect species interactions with a good level of confidence. These extracted relationships were then used to build ecological graphs that represent how species interact in nature.

Overall, the study showed that LLMs can be used to extract useful ecological information from scientific literature. The final pipeline was able to build interaction graphs from text in an automated way, without needing any rule-based systems or external taxonomic databases.

6.2 Main Contributions

This work led to the following key contributions:

- A complete pipeline was developed for extracting species interactions from scientific abstracts using LLMs.
- Two different approaches to NER were compared: fine-tuning and few-shot prompting. The results showed that few-shot prompting was more effective and required fewer resources.
- A relation extraction setup was created using few-shot prompting that focused on achieving high precision and a low error rate.
- The final pipeline was able to build ecological interaction graphs directly from text without using predefined rules or ontologies.
- Several limitations of using LLMs for ecological extraction were identified, such as handling vague terms, prompt instability, and hallucinations.
- The study also showed the value of prompt-based extraction for domains that lack labeled data or infrastructure for model fine-tuning.

6.3 Future Work

This project opened several directions for future improvement and research:

- Building a better dataset: One of the main challenges was the lack of a high-quality dataset for training or fine-tuning relation extraction models. In the future, creating an annotated dataset focused on ecological relationships could help improve performance and enable model training.
- **Fine-tuning larger models:** Due to limited infrastructure, it was not possible to fine-tune larger models like Llama-4-Scout-17B-16E-Instruct, even though it performed best during few-shot prompting. With better hardware or cloud resources, fine-tuning these models could further improve results.

- **Using taxonomic databases:** Adding taxonomic validation steps, such as checking detected species against GBIF, NCBI, or GloBI, could help reduce false positives and improve the quality of the extracted graphs.
- **Improving prompt stability:** Since the model's output was sensitive to prompt wording, future work could explore more stable prompt templates or instruction-tuned models that give more consistent results.
- **Combining with knowledge bases:** Integrating LLMs with external ecological databases like GloBI could help verify predictions and add missing information.
- **Scaling to full-text documents:** This thesis focused on abstracts, but full-text articles may provide richer information and more interaction examples. Scaling the system to handle larger texts is the next step.
- **Working with ecologists:** Collaborating with biodiversity experts could help improve label definitions, validate predictions, and ensure the system meets real ecological research needs.

A Code

NER pipeline using few-shot prompting. [Ahm25]

```
import os
import sys
import csv
import re
import json
from datetime import datetime
from typing import List, Dict, Any, Optional
import pandas as pd
from llmner import FewShotNer
from llmner.data import AnnotatedDocument, Annotation
from llmner.utils import annotated_document_to_conll

FEW_SHOT_ENTITIES = {
    "common_name": "The non-scientific name used to identify a species in
        everyday language. These names often describe physical
        characteristics, geographic origin, or behavior of the organism.",
    "scientific_name": "The formal taxonomic name for a species, including
        binomial nomenclature, abbreviated forms, genus names alone, species
        names in taxonomic context, and Latin taxonomic descriptors.
        Scientific names are the standardized nomenclature used in biology to
        refer to specific species."
}

FEW_SHOT_EXAMPLES = [
    AnnotatedDocument(
        text="The complete sequence of the 16S rRNA gene of Mycoplasma felis,
            isolated from cats, was determined.",
```

```
        annotations=[
            Annotation(start=53, end=68, label="scientific_name",
                text="Mycoplasma felis"),
            Annotation(start=80, end=84, label="common_name", text="cats"),
        ],
    ),
    AnnotatedDocument(
        text="Mouse interleukin-2 (IL-2) stimulated the proliferation of mouse
            and rat cells but human IL-2 stimulated rat cells more effectively
            than mouse cells.",
        annotations=[
            Annotation(start=0, end=5, label="common_name", text="Mouse"),
            Annotation(start=59, end=64, label="common_name", text="mouse"),
            Annotation(start=69, end=72, label="common_name", text="rat"),
            Annotation(start=99, end=102, label="common_name", text="rat"),
            Annotation(start=129, end=134, label="common_name", text="mouse"),
        ],
    ),
    ...
]

def chunk_text(text: str, max_tokens: int = 512, chunk_overlap: int = 0) ->
    List[str]:
    """
    Langchain-style recursive text splitter without dependencies.
    """
    def split_recursive(text, max_tokens):
        separators = ["\n\n", "\n", r"(?<=[.?!])\s", " "]
        for sep in separators:
            parts = re.split(sep, text) if not sep.startswith("(?") else
                re.split(sep, text)
            chunks, current = [], []
            total_words = 0
            for part in parts:
                word_count = len(part.split())
                if total_words + word_count <= max_tokens:
                    current.append(part)
```

```
                total_words += word_count
            else:
                if current:
                    chunks.append(" ".join(current).strip())
                current = [part]
                total_words = word_count
        if current:
            chunks.append(" ".join(current).strip())
        if all(len(chunk.split()) <= max_tokens for chunk in chunks):
            return chunks
    words = text.split()
    return [
        " ".join(words[i:i+max_tokens])
        for i in range(0, len(words), max_tokens)
    ]
base_chunks = split_recursive(text, max_tokens)
if chunk_overlap > 0:
    overlapped_chunks = []
    for i in range(0, len(base_chunks)):
        current_chunk = base_chunks[i]
        if i > 0:
            previous_chunk = base_chunks[i - 1]
            overlap_words = "
                ".join(previous_chunk.split()[-chunk_overlap:])
            current_chunk = f"{overlap_words} {current_chunk}"
        overlapped_chunks.append(current_chunk.strip())
    return overlapped_chunks
return base_chunks
def initialize_fewshot_ner(model: str = None, temperature: float = 0.1,
    prompting_method: str = "multi_turn") -> FewShotNer:
    """
    Initialize and contextualize the FewShotNer model.
    """
    if model is None:
        model = os.environ.get("LLMNER_MODEL",
            "meta-llama/Llama-4-Scout-17B-16E-Instruct")
    few_model = FewShotNer(
```

```
        model=model,
        temperature=temperature,
        prompting_method=prompting_method,
        final_message_with_all_entities=True
    )
    few_model.contextualize(entities=FEW_SHOT_ENTITIES,
        examples=FEW_SHOT_EXAMPLES)
    return few_model
def process_s800_abstracts(
    abstracts_dir: str,
    few_model: FewShotNer,
    chunk_size: int = 1024,
    chunk_overlap: int = 0
) -> (List[Dict[str, Any]], List[Dict[str, Any]]):
    """
    Process abstracts from S800 corpus with the given chunk size.
    Returns (results, conll_results)
    """
    results = []
    conll_results = []
    abstract_files = [f for f in os.listdir(abstracts_dir) if
        f.endswith('.txt')]
    for filename in abstract_files:
        doc_id = filename[:-4]
        if not allowed_docs or doc_id in allowed_docs:
            with open(os.path.join(abstracts_dir, filename), 'r',
                encoding='utf-8') as f:
                content = f.read()
            chunks = chunk_text(content, max_tokens=chunk_size,
                chunk_overlap=chunk_overlap)
            for i, chunk in enumerate(chunks):
                model_output = few_model.predict([chunk])
                annotations = model_output[0]
                conll_format = annotated_document_to_conll(annotations)
                results.append({
                    'file': filename,
                    'doc_id': doc_id,
```

```
                'content': content,
                'annotations': annotations.annotations,
            })
            conll_results.append({
                'doc_id': doc_id,
                'conll': conll_format
            })
    return results, conll_results
def save_results_to_csv(results: List[Dict[str, Any]], csv_path: str):
    """
    Save the results to a CSV file in the specified format.
    """
    fieldnames = ['file', 'doc_id', 'content', 'Annotation Text', 'start',
        'end', 'label']
    with open(csv_path, 'w', newline='', encoding='utf-8') as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        for result in results:
            base_info = {
                'file': result['file'],
                'doc_id': result['doc_id'],
                'content': result['content'],
            }
            if 'annotations' in result and result['annotations']:
                for annotation in result['annotations']:
                    row = base_info.copy()
                    try:
                        row['Annotation Text'] = annotation['text'] if 'text' in
                            annotation else ''
                        row['start'] = annotation['start'] if 'start' in
                            annotation else ''
                        row['end'] = annotation['end'] if 'end' in annotation
                            else ''
                        row['label'] = annotation['label'] if 'label' in
                            annotation else ''
                    except (TypeError, KeyError):
                        try:
```

```
                        row['Annotation Text'] = getattr(annotation, 'text',
                            '')
                        row['start'] = getattr(annotation, 'start', '')
                        row['end'] = getattr(annotation, 'end', '')
                        row['label'] = getattr(annotation, 'label', '')
                    except (AttributeError, TypeError):
                        row['Annotation Text'] = str(annotation)
                        row['start'] = ''
                        row['end'] = ''
                        row['label'] = ''
                writer.writerow(row)
        else:
            row = base_info.copy()
            row['Annotation Text'] = ''
            row['start'] = ''
            row['end'] = ''
            row['label'] = ''
            writer.writerow(row)
def main():
    """ Main function to run the pipeline as a script."""
    os.environ.setdefault("OPENAI_API_BASE")
    os.environ.setdefault("OPENAI_API_KEY")
    S800_DIR = "filepath"
    S800_ABSTRACTS_DIR = os.path.join(S800_DIR, "zoo")
    OUTPUT_CSV = "output.csv"

    few_model = initialize_fewshot_ner()
    results, conll_results = process_s800_abstracts(
        abstracts_dir=S800_ABSTRACTS_DIR,
        few_model=few_model,
        chunk_size=1024,
        chunk_overlap=0
    )
    save_results_to_csv(results, OUTPUT_CSV)
if __name__ == "__main__":
    main()
```

Listing A.1: Species Named Entity Recognition Pipeline

Relation Extraction: Species Interaction Few-Shot Prompting Pipeline

```
import os
import json
import pandas as pd
from tqdm import tqdm
import re
import logging
from langchain_core.prompts import PromptTemplate
from langchain_openai import ChatOpenAI

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s
    - %(message)s')
logger = logging.getLogger(__name__)

class SpeciesRelationshipExtractor:
    """Extract ecological relationships between species from text with NER
        results."""

    def __init__(self, model_name="meta-llama/Llama-3.3-70B-Instruct",
        temperature=0.1, batch_size=10,
                confidence_threshold=0.5, debug=False):
        self.model_name = model_name
        self.temperature = temperature
        self.batch_size = batch_size
        self.confidence_threshold = confidence_threshold
        self.debug = debug

        self.interaction_df = None
        self.interaction_details = {}
        self.interaction_to_iri = {}

        self.llm = ChatOpenAI(model_name=model_name, temperature=temperature)
        self.prompt_template = self._create_few_shot_prompt()

        logger.info(f"Initialized SpeciesRelationshipExtractor with model:
            {model_name}, temp: {temperature}")
```

```
    def _format_relationship_types(self):
        formatted = []
        for interaction, details in self.interaction_details.items():
            if 'source' in details and 'target' in details:
                description = f"{interaction}: {details['source']} →
                    {details['target']}"
            else:
                description = interaction
            formatted.append(f"- {description}")
        return "\n".join(formatted)

    def load_interaction_types(self, csv_path):
        try:
            df = pd.read_csv(csv_path)
            logger.info(f"Loaded interaction types with {len(df)} rows")
            interaction_details = {}
            interaction_to_iri = {}
            for _, row in df.iterrows():
                interaction = row['interaction']
                interaction_details[interaction] = {
                    'source': row['source'],
                    'target': row['target'],
                    'termIRI': row['termIRI']
                }
                if 'termIRI' in row:
                    interaction_to_iri[interaction] = row['termIRI']
            self.interaction_df = df
            self.interaction_details = interaction_details
            self.interaction_to_iri = interaction_to_iri
            return df, interaction_details, interaction_to_iri
        except Exception as e:
            logger.error(f"Error loading interaction types: {str(e)}")
            return None, {}, {}

    def _create_few_shot_prompt(self):
        template = """
You are an expert in extracting ecological relationships between species
```

```
    entities.
Your task is to identify whether there is a specific interaction between the
    annotated entities in the given text.

Only use the following interaction types, grouped by category:
.... """
        return PromptTemplate(input_variables=["text", "entities",
            "interaction_types"], template=template)

    def _passes_entity_filter(self, text):
        return True

    def prepare_document(self, doc):
        filtered_entities = []
        for i, entity in enumerate(doc.get("entities", [])):
            new_entity = entity.copy()
            new_entity["id"] = len(filtered_entities)
            filtered_entities.append(new_entity)
        entity_list = "\n".join([
            f"Entity ID {e['id']}: \"{e['text']}\" (Type: {e.get('label',
                'SPECIES')})"
            for e in filtered_entities
        ])
        return {
            "document_id": doc.get("document_id", "unknown"),
            "text": doc.get("text", ""),
            "original_entities": doc.get("entities", []),
            "filtered_entities": filtered_entities,
            "entity_list": entity_list
        }

    def extract_relationships(self, prepared_doc):
        if len(prepared_doc["filtered_entities"]) < 2:
            return []
        doc_id = prepared_doc.get("document_id", "unknown")
        prompt_inputs = {
            "interaction_types": self._format_relationship_types(),
```

```
            "text": prepared_doc["text"],
            "entities": prepared_doc["entity_list"]
        }
        prompt = self.prompt_template.format(**prompt_inputs)
        try:
            response = self.llm.invoke(prompt)
            response_content = getattr(response, 'content', '')
            if self.debug:
                logger.info(f"Response preview: {response_content[:200]}")
            relationships = []
            relationship_pattern =
                r'E1:\s*([^\n]+)\s*\nE2:\s*([^\n]+)\s*\nrelation:\s*E1\s+(\w+)\s+E2\s*\nconfi
            matches = re.findall(relationship_pattern, response_content)
            seen = set()
            for match in matches:
                source, target, rel, conf = map(str.strip, match)
                key = (source.lower(), target.lower(), rel)
                if key not in seen and float(conf) >= self.confidence_threshold:
                    seen.add(key)
                    relationships.append({
                        "source_text": source,
                        "target_text": target,
                        "relation": rel,
                        "confidence": float(conf)
                    })
            return relationships
        except Exception as e:
            logger.error(f"Error extracting relationships for doc {doc_id}:
                {str(e)}")
            return []

    def load_ner_results(self, ner_results_path):
        try:
            df = pd.read_csv(ner_results_path)
            documents = []
            for doc_id, group in df.groupby('doc_id'):
                content = group['content'].iloc[0] if 'content' in
```

```
                group.columns else ""
            entities = []
            for _, row in group.iterrows():
                if pd.notna(row.get('start')) and pd.notna(row.get('end'))
                    and pd.notna(row.get('Annotation Text')):
                    entity_text = row['Annotation Text']
                    if self._passes_entity_filter(entity_text):
                        entities.append({
                            'id': len(entities),
                            'text': entity_text,
                            'start': int(row['start']),
                            'end': int(row['end']),
                            'label': row.get('label', 'SPECIES')
                        })
            if entities:
                documents.append({
                    'document_id': doc_id,
                    'text': content,
                    'entities': entities
                })
        return documents
    except Exception as e:
        logger.error(f"Error loading NER results: {str(e)}")
        return []

def process_documents(self, documents, output_path=None):
    results = []
    for i in range(0, len(documents), self.batch_size):
        batch = documents[i:i+self.batch_size]
        for doc in tqdm(batch):
            prepared_doc = self.prepare_document(doc)
            relationships = self.extract_relationships(prepared_doc)
            doc["relationships"] = relationships
            results.append(doc)
    if output_path:
        with open(output_path, 'w', encoding='utf-8') as f:
            json.dump(results, f, indent=2)
```

```
        return results

    def run_pipeline(self, ner_results_path, output_path):
        documents = self.load_ner_results(ner_results_path)
        return self.process_documents(documents, output_path)
```

Listing A.2: Species Relationship Extraction Pipeline

B Math

To evaluate the performance of species recognition and interaction extraction, this study uses precision, recall, F1-score, and error rate. The focus is on maximizing recall in NER to capture all relevant species mentions and minimizing the error rate in RE to ensure accurate interactions. Missing species leads to incomplete graphs, while incorrect relations can misrepresent ecological patterns. The metrics are defined as follows:

- **Precision:** The proportion of correctly identified positive cases among all predicted positives:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

- **Recall:** The proportion of correctly identified positive cases among all actual positives:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

- **F1-score:** The harmonic mean of precision and recall:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Error Rate:** Used instead of accuracy due to the absence of true negatives in sequence labeling tasks. It reflects the proportion of incorrect predictions:

$$\text{Error Rate} = \frac{\text{False Positives (FP)} + \text{False Negatives (FN)}}{\text{Total Predictions}} = 1 - \frac{\text{True Positives (TP)}}{\text{Total Predictions}}$$

C Dataset

This study uses the **Species-800 (S800)** dataset to train and evaluate the species name recognition pipeline and pass it to the Few-shot prompting NER and RE pipeline. S800 is a high-quality, expert-annotated dataset designed for taxonomic named entity recognition. It was introduced as part of the LINNAEUS project and is used in biodiversity and biomedical text mining. The Species-800 dataset consists of **800 scientific abstracts**, divided into eight categories, each representing a specific subdomain of biology, including botany, zoology, and microbiology. Each abstract is manually annotated with species mentions using character-level offsets, allowing for precise entity recognition tasks.

Bibliography

[ALK23] Nora Abdelmageed, Felicitas Löffler, and Birgitta König-Ries. "BiodivBERT: a Pre-Trained Language Model for the Biodiversity Domain." In: *SWAT4HCLS*. 2023, pp. 62–71 (cit. on pp. 1, 65, 66).

[Abd+22] Nora Abdelmageed et al. "BiodivNERE: Gold standard corpora for named entity recognition and relation extraction in the biodiversity domain." In: *Biodiversity Data Journal* 10 (2022), e89481 (cit. on p. 1).

[Ahm25] Mahshid Ahmadi. *MahshidAhmadi Thesis Scripts*. `https://git.fim.uni-passau.de/ahmadim/MahshidAhmadi_Thesis_Scripts`. Accessed: 2025-07-08. 2025 (cit. on p. 73).

[AI 23] AI Research Studies. *History of Large Language Models: From 1940 to 2023*. 2023 (cit. on p. 9).

[Ant25a] Anthropic. *Claude AI Shared Conversation, Abstract and Conclusion Academic Structure Check*. Used only for grammar and structural checks. No contribution to content or analysis. Accessed July 2025. 2025. URL: `https://claude.ai/share/f843bb69-a44d-4b37-8408-e5d72345f9bf` (cit. on pp. iv, 70).

[Ant25b] Anthropic. *Claude AI Shared Conversation, Experiment Academic Structure Check*. Used only for grammar and structural checks. No contribution to content or analysis. Accessed July 2025. 2025. URL: `https://claude.ai/share/5c5b81fa-b516-4876-97c9-0789d036145b` (cit. on p. 40).

[Ant25c] Anthropic. *Claude AI Shared Conversation, Introduction and Background Academic Structure Check*. Used only for grammar and structural checks. No contribution to content or analysis. Accessed July 2025. 2025. URL: `https://claude.ai/share/770410e4-053d-4700-9823-ed5a3293eabb` (cit. on pp. 1, 4).

[Ant25d] Anthropic. *Claude AI Shared Conversation, Methodology and Discussion Academic Structure Check*. Used only for grammar and structural checks. No contribution to content or analysis. Accessed July 2025. 2025. URL: `https://claude.ai/share/9e42cff3-ab6a-4acf-9747-fea3b5b8798b` (cit. on pp. 28, 63).

[AV04] Grigoris Antoniou and Frank Van Harmelen. *A semantic web primer*. MIT press, 2004 (cit. on p. 25).

[Ant] D. Balani D Antoszyk. "Identifying Suspicious Ecological Interactions in the Global Biotic Interactions (GloBI) Database." In: (cit. on p. 3).

[ASE22] Mohsen Asghari, Daniel Sierra-Sosa, and Adel S Elmaghraby. "BINER: A low-cost biomedical named entity recognition." In: *Information Sciences* 602 (2022), pp. 184–200 (cit. on p. 24).

[AL23] Dhananjay Ashok and Zachary C Lipton. "Promptner: Prompting for named entity recognition." In: *arXiv preprint arXiv:2305.15444* (2023) (cit. on p. 16).

[Aue+07] Sören Auer et al. "Dbpedia: A nucleus for a web of open data." In: *international semantic web conference*. Springer. 2007, pp. 722–735 (cit. on p. 25).

[Bai+22] Yuntao Bai et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback." In: *arXiv preprint arXiv:2204.05862* (2022) (cit. on p. 13).

[Bas+03] Jordi Bascompte et al. "The nested assembly of plant–animal mutualistic networks." In: *Proceedings of the National Academy of Sciences* 100.16 (2003), pp. 9383–9387 (cit. on p. 21).

[Bec+14] Jan Beck et al. "Spatial bias in the GBIF database and its effect on modeling species' geographic distributions." In: *Ecological Informatics* 19 (2014), pp. 10–15 (cit. on p. 2).

[Bek+18] Giannis Bekoulis et al. "Joint entity recognition and relation extraction as a multi-head selection problem." In: *Expert Systems with Applications* 114 (2018), pp. 34–45 (cit. on p. 24).

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. "Scientific American article: The Semantic Web." In: *Scientific American issue* (2001) (cit. on p. 25).

[Bes+21] Tarek R Besold et al. "Neural-symbolic learning and reasoning: A survey and interpretation 1." In: *Neuro-symbolic artificial intelligence: The state of the art*. IOS press, 2021, pp. 1–51 (cit. on p. 27).

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006 (cit. on p. 5).

[Bol+08] Kurt Bollacker et al. "Freebase: a collaboratively created graph database for structuring human knowledge." In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, pp. 1247–1250 (cit. on p. 25).

[Bor+13] Antoine Bordes et al. "Translating embeddings for modeling multi-relational data." In: *Advances in neural information processing systems* 26 (2013) (cit. on p. 26).

[Bro+20] Tom Brown et al. "Language models are few-shot learners." In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901 (cit. on pp. 10, 12, 15).

[BLR23] Erik Brynjolfsson, Danielle Li, and Lindsey R Raymond. *Generative AI at Work*. Working Paper 31161. National Bureau of Economic Research, Apr. 2023 (cit. on p. 10).

[BA11] Laura A Burkle and Ruben Alarcón. "The future of plant–pollinator diversity: understanding interaction networks across time, space, and global change." In: *American journal of botany* 98.3 (2011), pp. 528–538 (cit. on p. 21).

[Cer+20] Jair Cervantes et al. "A comprehensive survey on support vector machine classification: Applications, challenges and trends." In: *Neurocomputing* 408 (2020), pp. 189–215 (cit. on p. 17).

[Cha23] Edward Y Chang. "Examining gpt-4: Capabilities, implications and future directions." In: *The 10th international conference on computational science and computational intelligence*. 2023 (cit. on pp. 13, 17).

[Che+21] Mark Chen et al. "Evaluating large language models trained on code." In: *arXiv preprint arXiv:2107.03374* (2021) (cit. on p. 18).

[Che+24] Xiaojun Chen et al. "A fine-grained self-adapting prompt learning approach for few-shot learning with pre-trained language models." In: *Knowledge-Based Systems* 299 (2024), p. 111968 (cit. on p. 41).

[Cho20] K. R. Chowdhary. "Natural Language Processing." In: *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, 2020, pp. 603–649 (cit. on p. 7).

[Chu+23] Michael Chui et al. "The state of AI in 2023: Generative AI's breakout year." In: (2023) (cit. on p. 6).

[Dan+22] Hai Dang et al. "How to prompt? Opportunities and challenges of zero- and few-shot learning for human-AI interaction in creative applications of generative models." In: *arXiv preprint arXiv:2209.01390* (2022) (cit. on p. 14).

[Dan+24] Gabriel Dansereau et al. "Overcoming the disconnect between interaction networks and biodiversity conservation and management." In: (2024) (cit. on p. iv).

[DQR22] Matheus L De Araujo, Adriano C Quaresma, and Flavio N Ramos. "GBIF information is not enough: national database improves the inventory completeness of Amazonian epiphytes." In: *Biodiversity and Conservation* 31.11 (2022), pp. 2797–2815 (cit. on p. 2).

[Dev+19] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186 (cit. on pp. 8, 11, 22, 26).

[Dia+21] Shizhe Diao et al. "Taming Pre-trained Language Models with N-gram Representations for Low-Resource Domain Adaptation." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021 (cit. on pp. 7, 9).

[Don+14] Xin Dong et al. "Knowledge vault: A web-scale approach to probabilistic knowledge fusion." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 601–610 (cit. on p. 26).

[Du+22] Nan Du et al. "Glam: Efficient scaling of language models with mixture-of-experts." In: *International conference on machine learning*. PMLR. 2022, pp. 5547–5569 (cit. on p. 12).

[Fab+23] Hermenegildo Fabregat et al. "Negation-based transfer learning for improving biomedical Named Entity Recognition and Relation Extraction." In: *Journal of Biomedical Informatics* 138 (2023), p. 104279 (cit. on p. 24).

[GNB10] Martin Gerner, Goran Nenadic, and Casey M Bergman. "LINNAEUS: a species name identification system for biomedical literature." In: *BMC bioinformatics* 11 (2010), pp. 1–17 (cit. on pp. 28, 65, 66).

[GG20] Benyamin Ghojogh and Ali Ghodsi. "Attention mechanism, transformers, BERT, and GPT: tutorial and survey." In: (2020) (cit. on p. 8).

[GHB09] Alec Go, Lei Huang, and Richa Bhayani. "Twitter sentiment analysis." In: (2009) (cit. on p. 8).

[GDO10] Ana M Martín González, Bo Dalsgaard, and Jens M Olesen. "Centrality measures and the importance of generalist species in pollination networks." In: *Ecological complexity* 7.1 (2010), pp. 36–43 (cit. on p. 22).

[Goo+16] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016 (cit. on p. 5).

[GS24] Nandita Goyal and Navdeep Singh. "Named Entity Recognition and Relationship Extraction for Biomedical Text: A comprehensive survey, recent advancements, and future research directions." In: *Neurocomputing* (2024), p. 129171 (cit. on pp. 23–25).

[Hab+17] Maryam Habibi et al. "Deep learning with word embeddings improves biomedical named entity recognition." In: *Bioinformatics* 33.14 (2017), pp. i37–i48 (cit. on p. 22).

[Had+23] Muhammad Usman Hadi et al. "A survey on large language models: Applications, challenges, limitations, and practical usage." In: *Authorea Preprints* (2023) (cit. on pp. 7, 13, 17).

[Him+17] Daniel Scott Himmelstein et al. "Systematic integration of biomedical knowledge prioritizes drugs for repurposing." In: *elife* 6 (2017), e26726 (cit. on p. 26).

[Hoc98] Sepp Hochreiter. "Recurrent Neural Net Learning and Vanishing Gradient." In: (Jan. 1998) (cit. on p. 8).

[Hof+22] Jordan Hoffmann et al. "Training compute-optimal large language models." In: *arXiv preprint arXiv:2203.15556* (2022) (cit. on p. 12).

[Hu+22] Edward J Hu et al. "Lora: Low-rank adaptation of large language models." In: *ICLR* 1.2 (2022), p. 3 (cit. on p. 35).

[Hu+23] Zhiqiang Hu et al. "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models." In: *arXiv preprint arXiv:2304.01933* (2023) (cit. on p. 14).

[Hua+23] Shaohan Huang et al. "Language is not all you need: Aligning perception with language models." In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 72096–72109 (cit. on p. 18).

[Hug24] Hugging Face. *Hugging Face: Machine Learning for Everyone*. Accessed: 2025-07-06. 2024 (cit. on p. 35).

[Ic24] Artifex Software Inc. and contributors. *PyMuPDF: Python bindings for MuPDF*. Accessed: 2025-07-06. 2024 (cit. on p. 32).

[IZK20] Vassilis N Ioannidis, Da Zheng, and George Karypis. "Few-shot link prediction via graph neural networks for covid-19 drug-repurposing." In: *arXiv preprint arXiv:2007.10261* (2020) (cit. on p. 26).

[Jia+23] Xuhui Jiang et al. "On the evolution of knowledge graphs: A survey and perspective." In: *arXiv preprint arXiv:2310.04835* (2023) (cit. on pp. 25–27).

[Jin+23] Hongye Jin et al. "Growlength: Accelerating llms pretraining by progressively growing training length." In: *arXiv preprint arXiv:2310.00576* (2023) (cit. on p. 7).

[Joa98] Thorsten Joachims. "Text categorization with support vector machines: Learning with many relevant features." In: *European conference on machine learning*. Springer. 1998, pp. 137–142 (cit. on p. 8).

[Kas+23] Enkelejda Kasneci et al. "ChatGPT for good? On opportunities and challenges of large language models for education." In: *Learning and individual differences* 103 (2023), p. 102274 (cit. on p. 18).

[Kel+24] Vipina K Keloth et al. "Advancing entity recognition in biomedicine via instruction tuning of large language models." In: *Bioinformatics* 40.4 (2024), btae163 (cit. on p. 41).

[Kim+21] Boseop Kim et al. "What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers." In: *arXiv preprint arXiv:2109.04650* (2021) (cit. on p. 17).

[Koj+22] Takeshi Kojima et al. "Large language models are zero-shot reasoners." In: *Advances in neural information processing systems* 35 (2022), pp. 22199–22213 (cit. on p. 10).

[Kom+24] Vamsi Krishna Kommineni et al. "Automating information retrieval from biodiversity literature using large language models: A case study." In: *Biodiversity Information Science and Standards* 8 (2024), e136735 (cit. on p. 1).

[KZP06] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. "Machine learning: a review of classification and combining techniques." In: *Artificial Intelligence Review* 26 (2006), pp. 159–190 (cit. on p. 17).

[KVK13] Jasper Kuperus, Cor J Veenman, and Maurice van Keulen. "Increasing NER recall with minimal precision loss." In: *2013 European Intelligence and Security Informatics Conference*. IEEE. 2013, pp. 106–111 (cit. on p. 45).

[Lee+20] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining." In: *Bioinformatics* 36.4 (2020), pp. 1234–1240 (cit. on pp. 23, 25, 65, 66).

[Lho+21] Quentin Lhoest et al. "Datasets: A community library for natural language processing." In: *arXiv preprint arXiv:2109.02846* (2021) (cit. on pp. 9, 12).

[Li+23a] Dongling Li et al. "Joint learning-based causal relation extraction from biomedical literature." In: *Journal of biomedical informatics* 139 (2023), p. 104318 (cit. on pp. 23, 25).

[Li+23b] Zongxi Li et al. "Label supervised llama finetuning." In: *arXiv preprint arXiv:2310.01208* (2023) (cit. on p. 35).

[Lie+21] Opher Lieber et al. "Jurassic-1: Technical details and evaluation." In: *White Paper. AI21 Labs* 1.9 (2021), pp. 1–17 (cit. on p. 12).

[Lin+22] Zeming Lin et al. "Language models of protein sequences at the scale of evolution enable accurate structure prediction." In: *BioRxiv* 2022 (2022), p. 500902 (cit. on p. 18).

[Liu+19] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach." In: *arXiv preprint arXiv:1907.11692* (2019) (cit. on pp. 9, 12).

[MV22] Shiva Mayahi and Marko Vidrih. "The impact of generative ai on the future of visual content marketing." In: *arXiv preprint arXiv:2211.12660* (2022) (cit. on p. 18).

[Mel+24] Pablo Hendrigo Alves de Melo et al. "A new R package to parse plant species occurrence records into unique collection events efficiently reduces data redundancy." In: *Scientific Reports* 14.1 (2024), p. 5450 (cit. on p. 1).

[Mik+10] Tomas Mikolov et al. "Recurrent neural network based language model." In: vol. 2. Sept. 2010, pp. 1045–1048 (cit. on p. 8).

[Mis24] Akanksha Mishra. "A Comprehensive Review of Artificial Intelligence and Machine Learning : Concepts, Trends, and Applications." In: *International Journal of Scientific Research in Science and Technology* 11 (Sept. 2024), pp. 126–142 (cit. on pp. 4, 5).

[MRC24] Fabricio Rios Montero, Ervin Rodríguez, and Maria Mora Cross. "Relation Extraction From Unstructured Species Descriptions Using TaxonNERD and LLaMA 2 7B." In: *Biodiversity Information Science and Standards* 8 (2024), e142382 (cit. on p. 2).

[MKL19] Gabriel Muñoz, W Daniel Kissling, and E Emiel van Loon. "Biodiversity Observations Miner: A web application to unlock primary biodiversity data from published literature." In: *Biodiversity data journal* 7 (2019), e28737 (cit. on p. 2).

[MDA21] Tendai Musvuugwa, Muxe Gladmond Dlomu, and Adekunle Adebowale. "Big data in biodiversity science: A framework for engagement." In: *Technologies* 9.3 (2021), p. 60 (cit. on p. 1).

[Néd+13] Claire Nédellec et al. "Overview of BioNLP shared task 2013." In: *Proceedings of the BioNLP shared task 2013 workshop*. 2013, pp. 1–7 (cit. on p. 29).

[NR13] M S Neethu and R Rajasree. "Sentiment analysis in twitter using machine learning techniques." In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. 2013, pp. 1–5 (cit. on p. 8).

[NGA19] Nhung TH Nguyen, Roselyn S Gabud, and Sophia Ananiadou. "COPIOUS: A gold standard corpus of named entities towards extracting species occurrence from biodiversity literature." In: *Biodiversity data journal* 7 (2019), e29626 (cit. on pp. 65, 66).

[Nic+15] Maximilian Nickel et al. "A review of relational machine learning for knowledge graphs." In: *Proceedings of the IEEE* 104.1 (2015), pp. 11–33 (cit. on p. 25).

[Ole+07] Jens M. Olesen et al. "The modularity of pollination networks." In: *Proceedings of the National Academy of Sciences* 104.50 (2007), pp. 19891–19896 (cit. on p. 21).

[Ope+23] J OpenAI Achiam et al. "GPT-4 technical report. arXiv." In: *arXiv preprint arXiv:2303.08774* (2023) (cit. on pp. 12, 17).

[Ort+23] Miguel Ortega-Martín et al. "Linguistic ambiguity analysis in ChatGPT." In: *arXiv preprint arXiv:2302.06426* (2023) (cit. on p. 58).

[OT+23] T Osawa, N Tsutsumida, et al. "The role of large language models in ecology and biodiversity conservation: Opportunities and Challenges." In: (2023) (cit. on p. 2).

[Ouy+22] Long Ouyang et al. "Training language models to follow instructions with human feedback." In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744 (cit. on p. 13).

[Paf+13a] Evangelos Pafilis et al. "The SPECIES and ORGANISMS Resources for Fast and Accurate Identification of Taxonomic Names in Text." In: *PLOS ONE* (2013) (cit. on p. 28).

[Paf+13b] Evangelos Pafilis et al. "The SPECIES and ORGANISMS resources for fast and accurate identification of taxonomic names in text." In: *PloS one* 8.6 (2013), e65390 (cit. on pp. 2, 29, 65, 66).

[Pen+23] Guilherme Penedo et al. "The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only." In: *arXiv preprint arXiv:2306.01116* (2023) (cit. on p. 12).

[PSM14] Jorrit H Poelen, James D Simons, and Chris J Mungall. "Global biotic interactions: An open infrastructure to share and analyze species-interaction datasets." In: *Ecological informatics* 24 (2014), pp. 148–159 (cit. on pp. 18, 20–22, 38, 66).

[Poi+21] Timothée Poisot et al. "Global knowledge gaps in species interaction networks data." In: *Journal of Biogeography* 48.7 (2021), pp. 1552–1563 (cit. on pp. iv, 2).

[Raf+20] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." In: *Journal of machine learning research* 21.140 (2020), pp. 1–67 (cit. on p. 12).

[Raw+22] Bhupesh Rawat et al. "Recent deep learning based nlp techniques for chatbot development: An exhaustive survey." In: *2022 10th International Conference on Cyber and IT Service Management (CITSM)*. IEEE. 2022, pp. 1–4 (cit. on p. 17).

[Ray23] Partha Pratim Ray. "ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope." In: *Internet of Things and Cyber-Physical Systems* 3 (2023), pp. 121–154. ISSN: 2667-3452 (cit. on p. 7).

[Ros00] R. Rosenfeld. "Two decades of statistical language modeling: where do we go from here?" In: *Proceedings of the IEEE* 88.8 (2000), pp. 1270–1278 (cit. on p. 8).

[RN16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. pearson, 2016 (cit. on p. 4).

[Sah+24] Pranab Sahoo et al. "A systematic survey of prompt engineering in large language models: Techniques and applications." In: *arXiv preprint* (2024) (cit. on pp. 14, 16).

[Sal23] Malik Sallam. "The utility of ChatGPT as an example of large language models in healthcare education, research and practice: Systematic review on the future perspectives and potential limitations." In: *MedRxiv* (2023), pp. 2023–02 (cit. on p. 18).

[SD03] Erik F Sang and Fien De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." In: *arXiv preprint cs/0306050* (2003) (cit. on p. 58).

[Sch23] Albrecht Schmidt. "Speeding Up the Engineering of Interactive Systems with Generative AI." In: *Companion Proceedings of the 2023 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '23 Com-

panion. Swansea, United Kingdom: Association for Computing Machinery, 2023, pp. 7–8. ISBN: 9798400702068 (cit. on p. 10).

[SBV98] M. Setnes, R. Babuska, and H.B. Verbruggen. "Rule-based modeling: precision and transparency." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28.1 (1998), pp. 165–169 (cit. on p. 8).

[Sil+17] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. arXiv: `1712.01815 [cs.AI]` (cit. on p. 4).

[Sin+12] Amit Singhal et al. "Introducing the knowledge graph: things, not strings." In: *Official google blog* 5.16 (2012), p. 3 (cit. on p. 25).

[SKW07] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge." In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 697–706 (cit. on p. 25).

[Sun+19] Zhiqing Sun et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." In: *arXiv preprint arXiv:1902.10197* (2019) (cit. on p. 26).

[Sun23] Zhongxiang Sun. "A short survey of viewing large language models in legal aspect." In: *arXiv preprint arXiv:2303.09136* (2023) (cit. on p. 18).

[Tou+23] Hugo Touvron et al. "Llama: Open and efficient foundation language models." In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on pp. 11, 12).

[Tro+16] Théo Trouillon et al. "Complex embeddings for simple link prediction." In: *International conference on machine learning*. PMLR. 2016, pp. 2071–2080 (cit. on p. 26).

[Vas+17a] Ashish Vaswani et al. "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 8).

[Vas+17b] Ashish Vaswani et al. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 6, 9, 10, 26).

[VA05] Diego Vázquez and Marcelo Aizen. "Community-wide patterns of specialization in plant-pollinator interactions revealed by null models." In: Feb. 2005, pp. 200–219 (cit. on pp. 20, 21).

[Váz+22] Diego P Vázquez et al. "Ecological interaction networks. What we know, what we don't, and why it matters." In: *Ecología Austral* 32.2 (2022), pp. 670–697 (cit. on p. 18).

[VMA24] Fabián Villena, Luis Miranda, and Claudio Aracena. "llmNER:(Zero| Few)-Shot Named Entity Recognition, Exploiting the Power of Large Language Models." In: *arXiv preprint arXiv:2406.04528* (2024) (cit. on pp. 34, 42).

[VK14] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase." In: *Communications of the ACM* 57.10 (2014), pp. 78–85 (cit. on p. 25).

[Wan+18] Hongwei Wang et al. "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems." In: *Proceedings of the 27th ACM international conference on information and knowledge management*. 2018, pp. 417–426 (cit. on p. 26).

[Wan+24] Sitong Wang et al. "ReelFramer: Human-AI co-creation for news-to-video translation." In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–20 (cit. on p. 18).

[Wan+19] Xiang Wang et al. "Kgat: Knowledge graph attention network for recommendation." In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 950–958 (cit. on p. 26).

[Wei+22a] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models." In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837 (cit. on p. 16).

[Wei+22b] Jason Wei et al. *Emergent Abilities of Large Language Models*. 2022. arXiv: 2206.07682 [cs.CL] (cit. on p. 7).

[Whi+23] Jules White et al. "A prompt pattern catalog to enhance prompt engineering with chatgpt." In: *arXiv preprint arXiv:2302.11382* (2023) (cit. on pp. 10, 14).

[Wol+20] Thomas Wolf et al. "Transformers: State-of-the-art natural language processing." In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 2020, pp. 38–45 (cit. on p. 35).

[Wu+25] Chaoyi Wu et al. "Towards evaluating and building versatile large language models for medicine." In: *npj Digital Medicine* 8.1 (2025), p. 58 (cit. on p. 41).

[Wu+23] Shijie Wu et al. "Bloomberggpt: A large language model for finance." In: *arXiv preprint arXiv:2303.17564* (2023) (cit. on pp. 7, 18).

[Xu+22] Frank F Xu et al. "A systematic evaluation of large language models of code." In: *Proceedings of the 6th ACM SIGPLAN international symposium on machine programming*. 2022, pp. 1–10 (cit. on p. 12).

[Yad+20] Shweta Yadav et al. "Relation extraction from biomedical and clinical text: Unified multitask learning framework." In: *IEEE/ACM transactions on computational biology and bioinformatics* 19.2 (2020), pp. 1105–1116 (cit. on p. 24).

[YB19] Vikas Yadav and Steven Bethard. "A survey on recent advances in named entity recognition from deep learning models." In: *arXiv preprint arXiv:1910.11470* (2019) (cit. on pp. 23, 25).

[Yan+14] Bishan Yang et al. "Embedding entities and relations for learning and inference in knowledge bases." In: *arXiv preprint arXiv:1412.6575* (2014) (cit. on p. 26).

[YP24] Ilker Yildirim and LA Paul. "From task structures to world models: what do LLMs know?" In: *Trends in Cognitive Sciences* (2024) (cit. on pp. 7, 13).

[Yin+17] Wenpeng Yin et al. "Comparative study of CNN and RNN for natural language processing." In: *arXiv preprint arXiv:1702.01923* (2017) (cit. on p. 17).

[ZSV14] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." In: *arXiv preprint arXiv:1409.2329* (2014) (cit. on p. 5).

[Zen+21] Wei Zeng et al. "PanGu-alpha: Large-scale autoregressive pretrained Chinese language models with auto-parallel computation." In: *arXiv preprint arXiv:2104.12369* (2021) (cit. on p. 10).

[Zha+23] Xinghua Zhang et al. "Wider and deeper llm networks are fairer llm evaluators." In: *arXiv preprint arXiv:2308.01862* (2023) (cit. on pp. 7, 13).

[Zha+19] Zhengyan Zhang et al. “ERNIE: Enhanced Representation through Knowledge Integration.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1441–1451 (cit. on p. 12).

[Zha+25] Xiaoyan Zhao et al. “Few-Shot Relation Extraction With Automatically Generated Prompts.” In: *IEEE Transactions on Neural Networks and Learning Systems* 36.3 (2025), pp. 4971–4983 (cit. on p. 14).

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie, dass ich die Masterarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Passau, July 8, 2025

_____

AUTHOR