INTERNATIONALE HOCHSCHULE

Bachelor Thesis

IU International University of Applied Sciences

Bachelor of Science Data Science

Performance improvement of Large Language Models for Named Entity Recognition using Ensemble Learning techniques

Sandro Schweiss
Matriculation Number: 3206640
Hans Bögl Gasse 18
2491 Neufeld

Supervisor: Dr. Bernhard von Guretzky
Submission Date: 22.10.2024

Abstract

This thesis explores the capabilities of ensemble learning methods, explicitly bagging and boosting, to mitigate hallucinations and improve the performance of LLMs in NER tasks. This work aims to contribute to the ongoing development of more reliable and accurate LLMs. The experiment evaluated these techniques using the DocRED dataset, comparing outputs generated by LLMs with and without ensemble learning. The results indicated that while both methods performed comparable to standard NER extraction, their impacts varied. Boosting reduced false negatives while bagging reduced false positives. However, the marginal overall performance gain of 0,7% came at the cost of significantly increased processing time and resource demands. Despite these limitations, the findings suggest that ensemble methods may be relevant in specialized use cases. Future research could focus on the applicability of these techniques across different language processing tasks or utilize the findings to formulate different approaches to mitigate hallucinations.

Diese Arbeit untersucht das Potenzial von Ensemble-Learning-Methoden, insbesondere Bagging und Boosting, um Halluzinationen zu verringern und die Leistung von LLMs in NER Aufgaben zu verbessern. Ziel dieser Arbeit ist es, zur fortlaufenden Entwicklung zuverlässigerer und präziserer LLMs beizutragen. Das Experiment bewertete diese Techniken anhand des DocRED-Datensatzes und verglich die Ergebnisse, die von LLMs mit und ohne ensemble learning generiert wurden. Die Ergebnisse zeigten, dass beide Methoden eine vergleichbare Leistung zur Standard-NER-Extraktion erzielten, jedoch mit unterschiedlichen Effekten. Boosting verringerte die Anzahl der falsch-negativen Ergebnisse, während Bagging hauptsächlich die falsch-positiven Ergebnisse reduzierte. Der marginale Leistungszuwachs von 0,7% ging jedoch mit einem erheblichen Anstieg der Verarbeitungszeit und des Ressourcenbedarfs einher. Trotz dieser Einschränkungen deuten die Ergebnisse darauf hin, dass ensemble Methoden in spezialisierten Anwendungsfällen relevant sein könnten. Zukünftige Forschung könnte sich auf die Anwendbarkeit dieser Techniken in anderen Sprachverarbeitungsaufgaben konzentrieren oder die Ergebnisse nutzen, um alternative Ansätze zur Verringerung von Halluzinationen zu entwickeln.

Keywords: Ensemble Learning, Bagging, Boosting, Large Language Models, Named Entity Recognition, Hallucinations, DocRED Dataset

Table of Contents

List of Figures

Figure Description **Page**

List of Abbreviations

Abbreviation	**Meaning**
NLP	Natural Language Processing
LLM	Large Language Model
NER	Named Entity Recognition
AI	Artificial Intelligence
LM	Language Model
ML	Machine Learning
GPU	Graphics Processing Unit
CPU	Central Processing Unit
SVM	Support Vector Machines
CRF	Conditional Random Fields
LSA	Latent Semantic Analysis
LDA	Latent Dirichlet Allocation
RNN	Recurrent Neural Networks
FFNN	Feedforward Neural Network
LSTM	Long Short-Term Memory
GRU	Gated recurrent units
CBOW	Continuous Bag of Words
GloVe	Global Vectors for Word Representation
ReLU	Rectified Linear Unit
GPT	Generative Pre-trained Transformer
BERT	Bidirectional Encoder Representations from Transformers
RAG	Retrieval-Augmented Generation
SaaS	Software-as-a-Service
TP	True Positives
FP	False Positives
TN	True Negatives
FN	False Negatives

1. Introduction

1.1 An Introduction to Large Language Models

Throughout history, the development and understanding of language has been pivotal in shaping how we interact and convey information. It serves as the most natural way of communication and expression, a user interface that everyone intuitively understands. Typically, systems and computers require their own set of commands or operations; however, Natural Language Processing (NLP) makes it possible for users to interact with systems using everyday language, enhancing accessibility and ease of use (Luo, 2023). While the field of NLP started in the 1950s, in its early years, the technology was limited to executing specific tasks, such as machine translation and named entity recognition. These tasks required tailoring algorithms to the individual problems, a far cry from mimicking human performance. As NLP advanced over the decades, its applications expanded, which enabled more sophisticated use cases. The first major breakthrough into the mainstream was Apple's integration of the voice assistant Siri in 2011. While still being fairly limited in its application, it sparked a global interest for corporations to invest in the field. Over the years, therefore, the models improved due to an increase in data, computing power, better algorithms, and more funding (Johri et al., 2021). Models have shifted to accommodate a substantial increase in their parameters and training data, leading to significant advancements in their capabilities across various NLP tasks. However, the pivotal moment for NLP came with the discovery and development of Transformer models. The combination of the transformer architecture with massive language models created the Large Language model (LLM), which revolutionized the field. The first major LLMs used by the wider public were OpenAI's GPT and Google's BERT. These LLMs have a more general and nuanced understanding of language, which makes them the backbone of many services and interfaces, such as chatbots, intelligent customer service systems, and many more. Over the years, these models have made extraordinary progress in a remarkably short time period. The state-of-the-art LLMs are the first systems that enable users to communicate naturally with computer systems. A plethora of potential applications and innovative technologies could be developed if not for its current major challenges (Naveed et al., 2024).

1.2 The Problems Holding Large Language Models back

Despite their impressive advancements, LLMs still grapple with significant problems, for which solutions are actively researched to this date. The major ones are biases, fake and toxic content, high computation costs, carbon footprint, and hallucinations. Skewed or imbalanced training data can result in outputs that are biased. The generation of fake and toxic content has the potential to impact individuals and the population negatively. The high computational costs and carbon footprint associated with training these large models have a considerable effect on the infrastructure and environment (Patil & Gudivada, 2024). One of the most critical concerns, however, is the phenomenon of hallucinations, where models generate believable sounding but false or nonsensical information. Numerous factors contribute to the occurrence of hallucinations, manifesting throughout different segments of the LLM life cycle, starting from the initial training and extending to the inference stage.

Even quality problems and issues in the data can further exacerbate the emergence of hallucinations. This makes hallucinations a deep-rooted and complex problem to solve, which holds LLMs back from being deployed in a large variety of critical industries (Tonmoy et al., 2024).

1.3 Consequences of Hallucinations

Numerous sectors, such as healthcare, law, and finance, have a high requirement for correct and accurate information for both processing and decision-making. In many scenarios of these industries, potential lives or livelihoods are at stake, necessitating a robust LLM that produces accurate and correct outputs rather than relying on chance. This constitutes such a significant problem that it currently renders LLMs for many potential sectors unusable. Given that LLMs are a general-purpose and versatile technology, they would not only affect systems in high-stakes industries. As a result of the increase in artificial intelligence (AI) interest, numerous sectors are presently incorporating LLMs into their business operations to enhance their efficiency and output. According to Eloundou and colleagues, around 19% of the jobs in the United States could have 50% of their tasks impacted by LLMs (Eloundou et al., 2023).

However, this massive adoption is not only limited to industries. Even a significant portion of the general public has adopted LLMs since the introduction of ChatGPT, and their impact can not be understated. According to the survey conducted by Similiarweb, ChatGPT gained an estimated 100 million active monthly users in only two months, which makes it the fastest-growing technology in all of history (Carr, 2023). Most consumers nowadays utilize LLMs primarily either as a means of information retrieval, such as a substitution for search engines, or to generate content, which could then be exposed to the internet. This presents a significant challenge when the data is inaccurate or nonsensical due to hallucinations. Misinformation on the internet, like social media, has always been a prevalent issue, but it has become progressively worse with the introduction of LLMs. AI has made it increasingly difficult to distinguish between authentic and generated content. The significance of this matter cannot be overstated, particularly given the fact that a significant proportion of the population depends on social media as their main source of news. Users who generate content rarely fact-check their information or are even malicious in spreading misinformation. This diminishes user trust in AI and leads to adverse outcomes in decision-making for a significant portion of the population (Rhys Leahy et al., 2021).

Due to this prevalent issue of hallucinations among major developers, the integration of LLMs across various industries has not yet occurred. Until a solution is found, the downsides will continue to be an impediment, and the potential innovations and advantages of this technology will remain untapped (Chiarello et al., 2024).

1.4 The Many Solutions to Hallucinations

The core question, therefore, remains: What are the possible solutions? Researchers and developers invested in the field have proposed multiple potential solutions over the years. These approaches can rougly be categorized into two major categories: model development and inference mitigation.

Model development involves adjustments and enhancements to the internal structure of an LLM, such as pre-training, fine-tuning, and the data in order to create more accurate and reliable models. On the other hand, inference focuses on improving model outputs through techniques such as prompt tuning or self-refinement methods to optimize the output as much as possible without changing the underlying architecture (Tonmoy et al., 2024). The most prominent organizations in the LLM space, like OpenAI and Google, invest more heavily in model development because advancements in model architecture and training can lead to broader, more significant improvements in AI capabilities. However, focusing on developing a new model is often impractical for individual researchers or smaller groups due to the immense costs and sophisticated infrastructure required. Training state-of-the-art models demands substantial computational power, access to vast amounts of data, and a team of experts to manage the processes involved in model design and optimization. Therefore, in addition to the limited access to the architecture of existing models, it is more feasible to utilize them as a foundation to focus on optimizing the inference (Zhang et al., 2023).

Consequently, this thesis seeks to investigate the potential of ensemble learning methods to mitigate hallucinations and improve the performance of LLMs. The objective of this research is to conduct an experiment aimed at evaluating the effectiveness of the ensemble learning techniques bagging and boosting, specifically for Named Entity Recognition (NER) tasks. By utilizing a pre-existing dataset, the aim is to quantify whether ensemble learning techniques can enhance the accuracy of the extracted data and reduce the hallucination rate. This is achieved by comparing the outputs generated by an LLM with and without ensemble learning techniques.

By examining this question, the thesis will add to existing studies by presenting real-world data on how ensemble learning can help address common issues like hallucinations and inaccuracies in LLMs. Additionally, this thesis seeks to provide practical insights and approaches to improve the applicability of LLMs in real-world settings.

1.5 Thesis Structure and Methodology

The thesis starts with an in-depth examination of existing literature, forming the foundation of the research. This section provides an exhaustive overview of existing and current literature on LLMs, establishing a basis for understanding the evolution of LLMs over time. It first delves into the overarching history of language models leading up to the inception of LLMs. This foundational exploration is essential for understanding the origins and development of LLMs in relation to earlier language models. This also clarifies the definition of an LLM and facilitates a better comprehension of its evolution. Subsequently, a thorough examination is undertaken into the internal history, development, and key milestones of these models using GPT as an example. This section explores the interconnected components of LLMs that work together to process and generate human-like text. Doing so aims to clarify how these models work and how they evolved. The next part delves further into the phenomenon of hallucinations, detailing its definition and potential consequences. Following this examination, types, causes, and strategies for reducing hallucinations are then discussed in detail.

Following the literature review, the thesis transitions into the actual experiment, where a comparative analysis of standard LLM approaches versus ensemble learning techniques is conducted. The initial step involves providing a comprehensive explanation of the experiment, outlining each essential element required for its execution. This includes the dataset used for testing, elaborating on the exact ensemble learning methods bagging and boosting, the LLM model chosen for the experiment, and how NER can be used to quantify the results. This step is then followed by the actual experiment, showcasing the exact procedure. This segment shows all steps from data pre-processing to the final outcome, which subsequently leads to the presentation of the results.

In the research findings section, the outputs of the different methodologies are compared and discussed using a designated test set. Statistical methods will evaluate performance metrics through accuracy, precision, recall, and F1 score. This evaluation aims to determine if ensemble learning methods significantly improve LLM performance in data extraction tasks. The results are also weighed against additional factors such as computational cost and scalability. This approach ensures that the chosen solution is not only technically sound but also practical and feasible in real-world applications. The results will then be analyzed, comparing their pros and cons. By evaluating all these aspects, a more robust and informed decision can be made to address the problem effectively. The section concludes with a discussion of the challenges encountered during the experiment, alongside the strategies used to overcome them.

In the conclusion segment, the thesis culminates by presenting a summary of the entire work, discussing the results, and offering critical reflections. This segment discusses the actual implications this thesis has on mitigation strategies, as well as the potential and limitations for future use cases. The section then concludes with a call for future research to build upon the discussed shortcomings of the experiment and new possibilities derived from these revelations.

2. Literature Analysis

The primary goal of the literature analysis is to establish a theoretical foundation that aids in comprehending how LLMs developed over time, their intricacies, and how hallucinations emerge, are classified, and are mitigated.

The literature review was conducted using credible databases like IEEE Xplore, arXiv, and Google Scholar, as well as journals such as Artificial Intelligence and AI Magazine. Initial searches were broad, using keywords like LLMs, ensemble methods, hallucinations, and NER. As the review progressed, more specific niche terms were employed to refine the search and align it with the focus of the thesis. The selection criteria for the literature were based on citation count, indicating the impact of the work and the overall relevance to the field. Foundational papers that introduced innovations transforming NLP and LLMs were prioritized in the early sections of the review. At the same time, more recent works were included to address cutting-edge topics, such as current hallucination mitigation and the latest developments in LLM models. The review intentionally encompassed a mix of technical, theoretical, and practical papers, ensuring a well-rounded perspective without bias toward any specific type of literature.

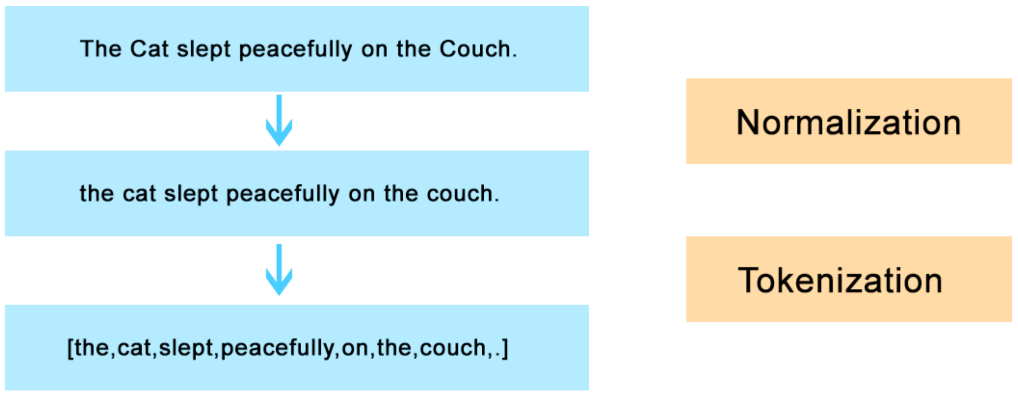
In terms of quality assessment, wherever possible, the methodology and diligence of each paper were evaluated to ensure reliability. This was particularly important when reviewing milestone papers that fundamentally altered the landscape of LLMs. However, in the case of contemporary papers, especially those covering the latest LLM models, peer review could not always be guaranteed due to the recency of publication. Thus, extra attention was paid to the research rationale and technical soundness. The literature included was selected with a rationale guided by a top-down strategy. As the understanding of the field deepened, gaps in knowledge were addressed, and specific questions that arose, such as the causes and types of hallucinations, as well as innovative approaches to their mitigation, were explored in greater detail. This iterative process ensured that the most relevant and impactful papers were included to support the thesis.

2.1 From LM to LLM

2.1.1 The Beginnings of NLP

To gain a comprehensive understanding of LLMs, it is crucial to explore their roots in the field of NLP. Although the first formal establishment of NLP dates back to the 1940s through Alan Turing's published paper, its roots can be linked to ancient philosophical concepts that explored the nature of human language. In its earliest inception, the concept of NLP was predominantly a philosophical debate discussed by early philosophers such as Aristotle, pondering the complexities of language and its use in communication (Hudry, 2015). Throughout the course of millennia, numerous prominent historical figures have embraced and shaped the formal examination of language and logic in their intellectual pursuits. What initially began as a philosophical discussion gradually transitioned towards mathematics and mechanisms during the Renaissance period. In the 17th century, Leibniz already proposed a theory for generating knowledge by utilizing systems and rules (Leibniz, 1989). It was not until the 20th century, centuries after their initial conception, that these ideas were revisited and reexamined. Linguists, mathematicians, logicians, and philosophers all contributed to the field with their unique perspectives and expertise. These foundational efforts set the stage for Turing's groundbreaking work that introduced the concept of AI. Turing's paper provided a theoretical framework for thinking about computation, language, and cognition, utilizing machines to process natural language. In this work, the Turing test was also proposed as a means to determine whether a machine's behavior is indistinguishable from that of a human. Through the adoption of the term intelligence in place of thinking, the discourse transitioned from ambiguity to a precise delineation (TURING, 1950). This led to the formal establishment of NLP as a sub-field of AI that utilizes computational systems to process human language inputs, comprehend them, and generate human-like text as a response (Ramanathan, 2024). The establishment of the field of NLP marked a significant turning point, denoting the beginning of a new age of projects and research endeavors aimed at understanding and manipulating human language through computational algorithms. Although the subfields of NLP were not formally established until decades later, linguists at the time employed a so-called structuralist approach to analyze language. They primarily focused on the inherent rules of language, emphasizing the relationships between letters, words, and sentences. This approach laid the foundation for the concepts of syntax, semantics, discourse, and pragmatics, which collectively form the domain of NLP (Bally & Sechehaye, 1986, p. 121). Syntax is the study of the structure and organization of words in a language, including grammar rules and sentence formation (Chomsky, 2020, p. 11). Semantic refers to the meaning behind these words and sentences, understanding the intended message (Wilks, 1975). Discourse examines the organization of sentences and phrases to create coherent communication across extended text passages (Grosz & Candace, 1986). Pragmatics, the final sub-field, focuses on how language is used within different contexts and the resulting implications (Allen, 1995, pp. 391–465). Based on this framework, early approaches focused on rule-based methods and linguistic theories to process natural language data, leading to the creation of the first rule-based language models. ELIZA, developed in the 1960s, was one of the earliest examples. It was created by Joseph Weizenbaum and mimicked a Rogerian psychotherapist by using simple pattern-matching techniques to respond to users' input (Weizenbaum, 1966). These early models also utilized the earliest forms of text normalization and tokenization techniques, which laid the foundation for processing textual data in a structured and standardized manner. Text normalization involves transforming a corpus into a consistent format by removing noise, standardizing spelling variations, and addressing other linguistic irregularities. Tokenization, on the other hand, entails breaking down text into smaller units called tokens. Depending on the particular task, these tokens might consist of words, phrases, or even individual characters. Through these processes, raw textual data is converted into a usable format (Jurafsky & Martin, 2024, pp. 4–29).

Fig. 1: Process of Normalization and Tokenization using textual data


The Cat slept peacefully on the Couch.
the cat slept peacefully on the couch.
[the,cat,slept,peacefully,on,the,couch,.]
Normalization
Tokenization

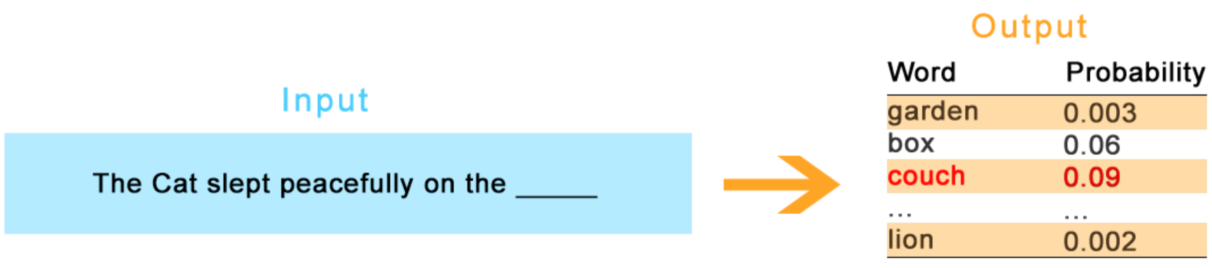
Source: Adopted from: (Wonhyeong, 2022)

Despite being revolutionary, these language models had severe limitations. They could not comprehend language in a meaningful way; rules had to be hand-written, they could not handle context over a conversation, and they were generally too limited (Grishman, 1986, pp. 159–171). For these reasons, subsequent language models have moved away from using rule-based algorithms but still rely on these foundational linguistic principles. It was not until their subsequent algorithmic advancement that a significant transformation occurred in the field of NLP, thereby concluding the era of rationalist NLP (Kaddari et al., 2021).

2.1.2 The Algorithmic Renaissance

In the evolution of NLP over the decades, a notable shift occurred away from linguistic-based methodologies towards a more algorithmic and mathematical approach. This transition marked the inception of what has been termed the empiricist wave within NLP. The empiricist wave denotes the movement from traditional rule-based strategies to methodologies that prioritize data-driven solutions and statistical models. The advancement was facilitated by the implementation of more powerful computers and the introduction of machine learning (ML). Consequently, this transition led to the creation of the first small language models (LM), which are models that generate text by forecasting the subsequent word based on assigning a probability (Kaddari et al., 2021). These smaller LMs shape the fundamental basis for how current LLMs operate. However, the need for text standardization remained important, bridging the advancements of the rationalist wave with empirical approaches. The models underwent considerable modifications in their underlying algorithms. By employing contemporary ML methodologies available at that time, these models were able to discern patterns and structures directly from the data. To facilitate this learning process, they required large, annotated corpora that offered detailed examples. Feature extraction played a pivotal role into then transforming standardized text into usable components, such as n-grams, which capture sequences of words and part-of-speech tags that categorize each word's grammatical role. These features enabled models to recognize linguistic patterns. Probabilistic models like Hidden Markov Models and Naive Bayes were then used to estimate probabilities of word sequences. To evaluate their efficacy, cross-validation, and performance indicators, including precision, recall, and accuracy, were applied, facilitating the refinement and optimization of the models as needed (Jurafsky & Martin, 2024, pp. 32–81).

Fig. 2: Example of LM predicting “couch” as the next word in a sentence



The Cat slept peacefully on the _____

Word	Probability
garden	0.003
box	0.06
couch	0.09
…	…
lion	0.002

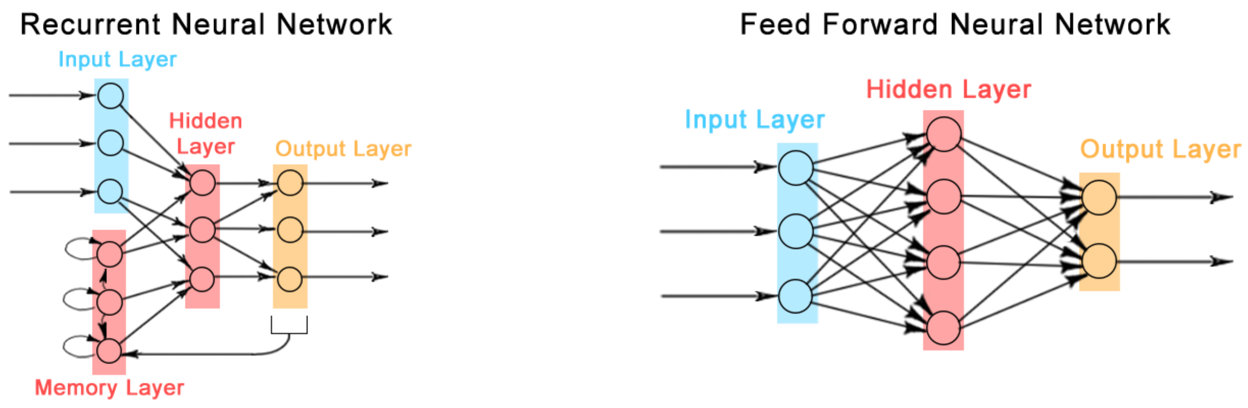
Source: Adopted from: (Jurafsky & Martin, 2024, p. 42)

Initially, many of these models relied heavily on multilingual textual corpora sourced from government datasets for supervised learning tasks. This approach generally led to better results but was limited by the availability of labeled data. With the growth of information on the World Wide Web, researchers recognized the potential of the massive amount of text data available on the Internet. This catalyzed a transition from traditional supervised learning methods to semi-supervised and unsupervised learning (Banko & Brill, 2001). This alternative approach produced suboptimal results when considering the same quantity of training data compared to supervised methods, but this could be offset by utilizing a larger amount of training data, a trend that has only begun to gain momentum since that time (Brill & Raymond, 1997). The year 1999 also marked a significant moment in computational capabilities with the introduction of Graphics Processing Units (GPUs). Prior to the advent of GPUs, Central Processing Units (CPUs) were predominantly used for various computational workloads. GPUs are engineered to handle multiple operations simultaneously due to their parallel processing architecture. The transition from CPUs to GPUs facilitated substantially faster processing times and increased throughput, which was crucial for training on larger datasets (Raina et al., 2009). Furthermore, building upon these recent innovations, the evolution of both statistical techniques and ML methodologies over time has resulted in significant developments such as the use of Support Vector Machines (SVMs) for text classification, the adoption of Conditional Random Fields (CRFs) for sequence labeling tasks, the development of Latent Semantic Analysis (LSA) together with Latent Dirichlet Allocation (LDA) for uncovering latent structures and topics within documents, and the establishment of statistical machine translation systems which significantly improved the quality of automatic translations (Jurafsky & Martin, 2024, pp. 105–186). Despite these improvements being significant at the time, they were not without their limitations. These LMs frequently encountered difficulties in understanding the deeper semantic meaning, primarily due to their reliance on statistical patterns and the necessity for extensive feature engineering. As a result, they were less adept at comprehending context and nuance. Furthermore, they faced challenges in managing relationships over an extended text corpus, which is essential for understanding meaning holistically for paragraphs and documents. These severe limitations would not be addressed until the next significant evolution of algorithms emerged (Fathi & Maleki Shoja, 2018).

2.1.3 The Rise of Deep Learning

The field of Deep Learning saw the emergence of its first algorithms already in 1986, marked by the inception of recurrent neural networks (RNN) (Rumelhart et al., 1986). These deep learning models differed from their statistical counterparts in how they learn and process data. RNNs use neural networks with multiple layers to autonomously identify patterns and connections in the data. The essence of these multi-layer neural networks is that they consist of several layers of interconnected nodes or neurons. Each layer transforms the input data through a series of operations involving weights, biases, and activation functions. Initially, the network starts with random weights. As information moves through every layer, the weights are modified to reduce the discrepancy between the predicted results and the true target values. They achieve this through a process known as backpropagation, where the network adjusts its parameters based on the errors it makes during training using the gradient descent algorithm. This process requires determining the gradients of the loss function for every weight, which indicates how each weight should be changed to reduce the prediction error. Through a process of repeatedly modifying the weights based on these gradients, the network iteratively improves its ability to make precise predictions and recognize patterns within the data. In an RNN, specifically, the network is designed to handle sequential data by maintaining a form of memory through its recurrent connections, called a hidden state. In contrast to the feedforward neural networks (FFNN) employed in current LLMs, which transmit information in a single direction, RNNs feature loops that enable the transfer of information over multiple time steps. This means that RNNs can leverage previous inputs to inform the processing of new inputs (Goodfellow et al., 2016, pp. 367–415).

Fig. 3: Architecture comparison of a RNN against a FFNN

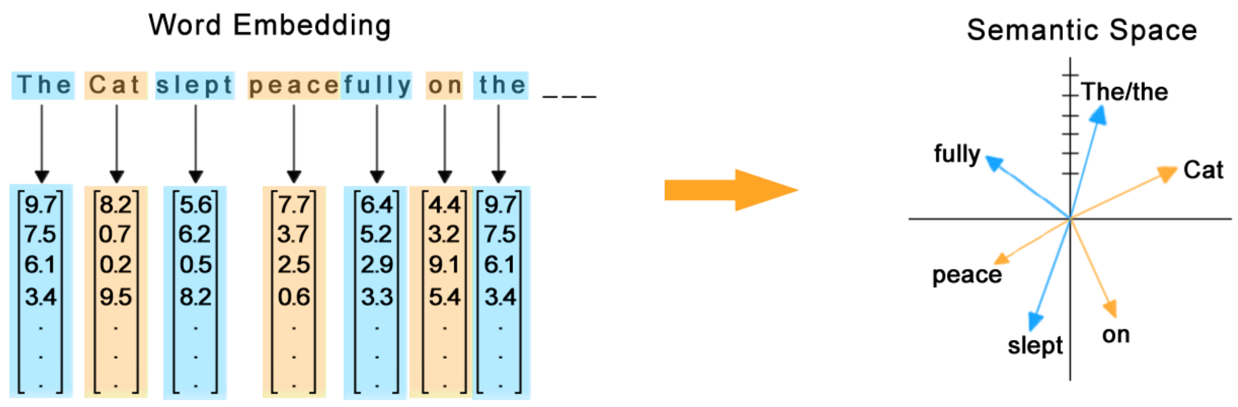


Sources: Adopted from: (Jordan, 1997), (McCulloch & Pitts, 1943)

In contrast, statistical models relied on explicit statistical rules and probabilities derived from data with fixed mathematical formulations. They often required manually designed features and did not adapt as flexibly to new patterns in the data (Fathi & Maleki Shoja, 2018). During the early stages of their development, algorithms like RNNs faced limited adoption due to challenges in training and their relative ineffectiveness, given the data and computational resources available at that time. Consequently, deep learning models were not initially prevalent; however, they underwent ongoing development similar to traditional statistical models. In 1997, the Long Short-Term Memory (LSTM) model was introduced, allowing for better handling of long-term word dependencies and overcoming problems with the original RNN and empirical approaches (Goodfellow et al., 2016, pp. 404–407). During the early 2000s, there was a notable rise in the utilization of deep learning models within the field of NLP, and approximately in 2007, LSTM models started to demonstrate superior performance over the statistical methods that were employed for specific tasks during that period (Fernández et al., 2007). Although these algorithms were already powerful on their own, their true potential was not fully realized until 2013, with the introduction of word embeddings. The emergence and integration of word embeddings represented a transformative shift in how linguistic data could be represented and processed, enabling neural networks to significantly surpass previous methodologies. Prior to this innovation, traditional methods of representing words relied heavily on sparse, high-dimensional vectors known as one-hot encodings, where each word was represented as a unique, binary vector.

Word embeddings are dense, low-dimensional vectors where each word in a vocabulary is mapped to a point in a continuous vector space. These vectors are learned from large text corpora and are designed to capture semantic meaning by positioning similar words close to each other in this high-dimensional space (Camacho-Collados & Pilehvar, 2020, pp. 27–42).

Fig. 4: Word embedding process: converting tokens into vectors, mapped to a 2D semantic space



Source: Adopted from: (Jurafsky & Martin, 2024, p. 113)

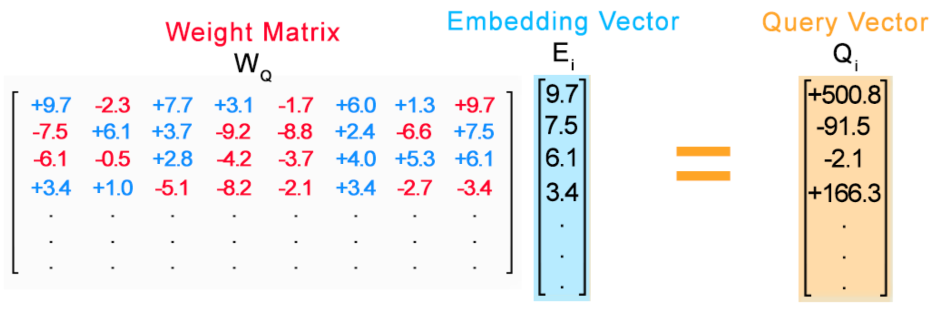
The initial method that employed word embeddings is known as Word2Vec, which was presented by Mikolov and colleagues. Google was the pioneering organization to successfully integrate these two approaches, leveraging neural networks to discern semantic relationships among words via un-supervised learning methods. This strategy enhanced the capability of algorithms to comprehend and produce human language. Based on this architecture, two key models emerged: Continuous Bag of Words (CBOW) and Skip-gram. These models gained significant traction due to their effectiveness and capacity to generate superior-quality embeddings (Mikolov et al., 2013). Then, Global Vectors for Word Representation (GloVe), developed by Pennington and colleagues, further improved word embeddings by leveraging both local context and global statistical information from the entire corpus. This method addressed some limitations of Word2Vec by incorporating global co-occurrence matrices (Pennington et al., 2014). In this short period, neural networks would also see many new architectural advancements. Gated recurrent units (GRUs) replaced LSTM models in 2014 as a simpler alternative to LSTMs with similar performance benefits but fewer parameters (Cho et al., 2014). In addition, the development of Seq2Seq models would significantly improve tasks like machine translation by enabling end-to-end training using an encoder-decoder architecture. The encoder processed and compressed the input into a single, summarized vector, which captured all essential information, and the decoder used this to generate the output sequence. In the context of machine translation, this output would correspond to the translated sentence in the desired target language (Sutskever et al., 2014).

After all these developments in the realm of deep learning, around 2015, neural networks effectively completely replaced statistical approaches. What remained from the empirical era is the general data-driven approach and the probabilistic mechanism of LMs predicting the next word in a sequence. Through neural networks, these advanced models gained the ability to understand more intricate patterns and connections within data. They moved beyond manually crafted features to automatically discover and represent the subtleties of language in dense, continuous vector spaces, allowing models to grasp context better. Despite these advancements, deep-learning language models still struggled with capturing long-range dependencies and context in sequences. They had difficulty understanding relationships between distant words, leading to performance issues in machine translation and text generation tasks. (Lipton et al., 2015). At this point, a paper introduced another significant advancement into the realm of deep learning: the attention mechanism (Bahdanau, Cho, & Bengio, 2014).

2.1.4 The Attention Mechanism

Introduced by Bahdanau and colleagues, the attention mechanism was initially developed as an extension of the encoder-decoder architecture. As previously elaborated, the encoder architecture condenses the input into a single fixed-size vector encapsulating all critical information. This, however, was simultaneously a major limitation. The models struggled with prolonged sentences, particularly those that exceeded the length of the sentences found within the training corpus. As a result, the decoder often struggled to capture long-range dependencies and contextual relationships accurately. The attention mechanism effectively solved this problem by dynamically focusing on tokens in the input sentence containing the most essential information. This was a significant improvement compared to the traditional encoder-decoder architecture, which treated all information as equal importance. The attention mechanism accomplishes this by allocating attention scores to the embeddings of words. This process is facilitated through three core components: Query, Key, and Value. The Query step in the process performs a matrix multiplication of weights, which were learned during the training process with the vector of the specific word embedding, resulting in query vectors (Bahdanau et al., 2014).

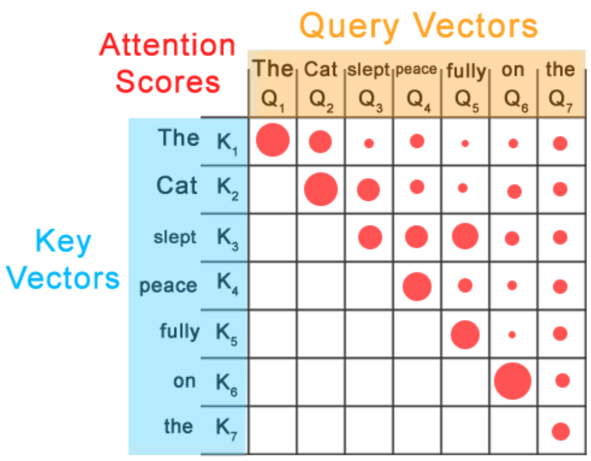
Fig. 5: Calculation of query vector via matrix multiplication of weights and word embeddings



Source: Adopted from: (Bahdanau et al., 2014)

Similarly, the Key step uses another weight matrix to create key vectors. The query vectors are then compared with each key vector using a dot product. The key vector refers to the embeddings of previous tokens that “attend to” the embedding of the query vector, meaning that these are words that give context to subsequent words of importance. This comparison determines how much focus each part of the input should receive, with the resulting attention scores indicating which pieces of information are most significant (Bahdanau et al., 2014).

Fig. 6: Calculation of attention score via dot product calculation of query and key vectors



Source: Adopted from: (Bahdanau et al., 2014)

In the subsequent phase, the value step is used to generate value vectors. These vectors are constructed from the same word embeddings utilized for the query and key vectors but are transformed through a separate learned weight matrix. After the attention scores are determined, the dot product between the Query and Key vectors is calculated. These scores are then scaled down by the square root of the Key vector's dimensionality. A softmax function is then applied to transform these scores into a probability distribution that spans from 0 to 1. Finally, these resulting probabilities are multiplied by their corresponding value vectors (Bahdanau et al., 2014).

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

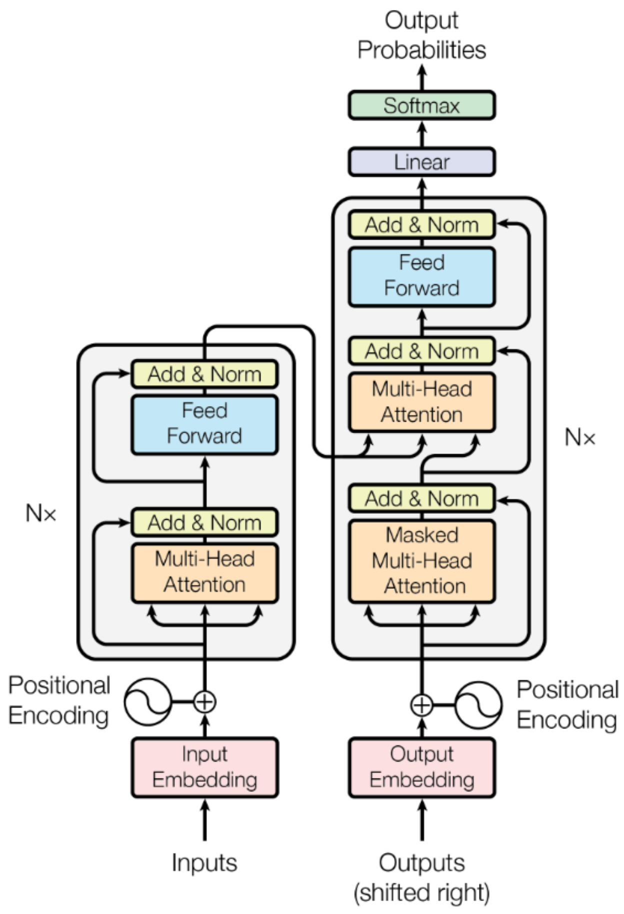
The weighted sum of these resulting values is then added to the word's original embedding vector, effectively determining what modifications should be made to reflect its contextual surroundings. This process culminates in forming a final context vector that encapsulates relevant information from the input sequence, which is subsequently utilized for output generation (1). This process is called a single head of attention (Bahdanau et al., 2014).

This mechanism allows for selective focus on essential parts of data while ignoring irrelevant information. Consequently, it enhances performance across various tasks by facilitating a more profound comprehension and interpretation of both short-term and long-term contexts in datasets. Following the introduction of this paper, attention mechanisms markedly enhanced the capabilities of existing models upon their application, such as the Seq2Seq model (Bahdanau et al., 2014). Most notably, however, it established the groundwork for the subsequent architecture that ultimately led to the creation of LLMs, the transformer (Vaswani et al., 2017).

2.1.5 The Transformer Revolution

In 2017, researchers from Google published the now-famous paper “Attention is All You Need”. This research expanded upon the concept of the attention mechanism by using it as its core component and presented the transformer, a new deep learning architecture. Prior to the introduction of transformers, RNNs, such as LSTMs and GRUs, operated on word sequences in a sequential fashion. This sequential processing was inherently time-consuming and posed significant challenges for parallelization. Furthermore, RNN architectures faced difficulties in effectively managing long sequences and accurately capturing long-range dependencies. The introduction of the transformer addressed these limitations through the implementation of self-attention mechanisms in the architecture itself. Self-attention allows the model to consider all positions in the input sequence simultaneously, calculating the relationships between tokens regardless of their positions. Therefore enabling models to process sequences of data in parallel rather than sequentially. This paradigm shift led to much quicker training times and enhanced performance in translation and text generation tasks, ultimately rendering the RNN architecture obsolete (Vaswani et al., 2017).

Fig. 7: The Transformer architecture



Source: (Vaswani et al., 2017)

The way transformer function is similar to previous models since it incorporates many previously established concepts. The process begins again with tokenization, where the input text is divided into smaller units. Subsequently, every token is mapped to a vector in a high-dimensional space, like in Figure 4, using the embedding layer, creating a sequence of vectors representing the input tokens. Since transformers do not have inherent positional awareness due to not being sequential like RNNs, positional encoding is added to these input embeddings. This step provides information about the position of each token in the sequence, which also changes the vector of the word embeddings. After the transformation into embeddings, they are fed into the transformer (Vaswani et al., 2017).

At its core, the transformer consists of two main components: the encoder and the decoder, each built from multiple repeating layers. The primary role of the encoder is to handle the input sequence. Each layer within the encoder has two main components: the self-attention mechanism and the feed-forward network. As previously elaborated, the self-attention mechanism calculates the attention scores using query (Q), key (K), and value (V) matrices, enabling them to concentrate on various segments of the input sequence, referred to as a single attention head. One of the key innovations introduced with transformers is multi-head attention. Multi-head attention involves performing several self-attention operations in parallel, each with different learned projections of Q, K, and V. By having multiple heads, the model can capture a richer set of relationships and nuances in the data. Each head processes the input slightly differently and independently (Vaswani et al., 2017).

Once the processing is complete, the outputs from all the heads are combined and then subjected to a linear transformation to generate the final result. This combined output contains information from all the different perspectives, allowing the model to understand the input more comprehensively. Every encoder layer contains a feed-forward neural network, as illustrated in Figure 3. This network performs two linear transformations, utilizing a Rectified Linear Unit (ReLU) activation function following the output of the self-attention mechanism. This step helps the model to learn and represent more intricate and abstract features from the input data, enhancing the model's ability to understand and generate language. After this step, residual connections and layer normalization are applied, which help the model learn more effectively and prevent issues like the vanishing gradient problem, where the gradients become too small to make meaningful changes as they pass through many layers. These three processes constitute a single layer, while a transformer generally comprises between six and twelve such layers. The decoder then processes the target sequence and generates the output. It operates similarly to the encoder, utilizing between six and twelve layers, consisting of four sub-components: masked self-attention, encoder-decoder attention, a feed-forward network, and residual and normalization. The masked self-attention mechanism is similar to the encoder's self-attention. However, it includes a mask to prevent the model from attending to future tokens in the sequence, ensuring that the prediction for a particular position depends only on known outputs at earlier positions. The attention mechanism between the encoder and decoder enables the decoder to focus on the pertinent sections of the input sequence by taking the encoder's output as keys and values and the decoder's own output as queries. This mechanism uses the encoder's outputs to understand the input text better and guide the output text's generation. The encoder's outputs help the decoder focus on the pertinent elements of the input sequence when predicting each token in the target sequence. Then, the feed-forward network, residual connections, and layer normalization are applied, as previously elaborated. Finally, the output from the decoder undergoes a linear transformation before being processed by a softmax layer to generate the probability distribution over the vocabulary for the next token prediction (Vaswani et al., 2017).

The transformer fundamentally altered how models interpret and generate human language, significantly enhancing their comprehension capabilities. One of the primary strengths of transformers is their ability to model relationships between words within a sequence without being constrained by their positional distances. This characteristic allows for a more nuanced understanding of context, leading to improved performance in various linguistic tasks. Additionally, transformers excel at managing large datasets due to their parallel processing capabilities. Unlike traditional sequential models, which process inputs one step at a time, transformers can analyze multiple elements simultaneously. This parallelism accelerates training and enables the utilization of extensive corpora that were previously impractical for other architectures. The transformer represented a novel architectural framework capable of processing information holistically and simultaneously, enabling it to process this information into predictions without the necessity of maintaining a hidden memory state (S. Islam et al., 2024). Nevertheless, despite its groundbreaking design, the original paper remained tethered to traditional methodologies, focusing predominantly on supervised translation tasks. The architecture was not conceived as a general-purpose system capable of addressing a diverse array of tasks (Vaswani et al., 2017). In contrast, researchers at OpenAI recognized the potential of this framework and applied it to the problem of next-word prediction on an unprecedented scale. This endeavor ultimately culminated in the creation of their iconic model, GPT (Radford & Narasimhan, 2018).

2.2 The Era of Large Language Models

2.2.1 What is a GPT?

The debut of the Generative Pre-trained Transformer (GPT) by OpenAI in 2018 marked, again, a significant milestone in the field of NLP. The term generative in GPT refers to its ability to create new text based on the patterns it has learned during training. The first GPT model was designed to tackle the problem of next-word prediction on an unprecedented scale, capable of processing hundreds of words of input or context at a time. In contrast to previous models that were constrained to specific tasks such as supervised translation, GPT was conceived as a more general system. Unlike purely discriminative models, which aim to classify or predict outcomes based on pre-existing information, GPT is designed to generate new content. This means that when given a specific input or question to the model, also known as a prompt, GPT can continue the text coherently and contextually relevantly. This generative aspect is achieved through pre-training. Pre-training is a critical concept of modern LMs, where the model is trained on a vast dataset in an unsupervised manner. During this phase, the model learns to understand the structure and nuances of language by predicting the next word in countless sentences. This process allows the model to develop a broad understanding of grammar, facts about the world, and some degree of reasoning ability. OpenAI trained the model on a, at the time, massive corpus of internet text, including 7000 books from a variety of domains. This, on the other hand, is made possible through the transformer. The transformer in GPT refers to the neural network architecture that underpins the model, which was elaborated in the previous chapter. GPT's pre-training on diverse text gave it a significant advantage: it could generate coherent and contextually appropriate text based on the patterns it had learned. This generative capability made GPT a versatile tool for a wide array of NLP tasks, such as language translation, summarization, and question answering, not needing to be present in the data, known as "Zero-shot" learning. GPT also outperformed in "One-shot" learning, where the model uses just one example to understand and complete a task, and "Few-shot" learning, which involves a few examples to grasp the task better and generate accurate results. After the initial phase of unsupervised training, a phase of supervised fine-tuning can follow. Fine-tuning involves taking the pre-trained model and then training it further on a smaller, task-specific dataset. This additional training helps the model adapt its general knowledge to perform better on the target task, like text classification or translation. It allows the model to adapt and optimize its learned representations for particular applications, leading to even better performance (Radford & Narasimhan, 2018).

Upon its initial launch, GPT-1 represented the most extensive model in existence, featuring 117 million parameters. This size led to the ability to generate text completions and understand the context to a certain extent. However, despite its groundbreaking architecture, GPT-1 was limited in terms of coherence and relevance when dealing with complex prompts or long-form content. Nonetheless, following the insights gained from GPT-1, subsequent developments would further expand the capabilities and scale of language models, which marked a crucial shift towards the era of LLMs and the development of models like GPT-2 and BERT (Radford & Narasimhan, 2018).

2.2.2 Transition towards LLMs

A year after this experiment, in 2019, OpenAI published a paper that presented GPT-2. Building on the foundation established by its predecessor, GPT-1, this model employed the same underlying approach but incorporated significant enhancements. One of these advancements was the scale of GPT-2, which represented a tenfold increase compared to GPT-1. The training dataset comprised approximately 8 million web pages, vastly outpacing the previous dataset. Additionally, GPT-2 boasted an, at the time, impressive architecture with 300 000 neurons and 1,5 billion parameters. It again underwent evaluation on various tasks, including reading comprehension, summarization, translation, and question answering. OpenAI's strategy of scaling up the model directly correlated with improved performance outcomes in the test results, significantly outperforming its predecessor. This approach of further increasing the computation power, training data, and network size initiated a trend that has persisted over subsequent years and signified a pivotal transition towards what we presently recognize as LLMs (Radford et al., 2019).

Around the release of GPT-2, other significant LLMs also emerged, each contributing uniquely to the advancement of NLP. Among these, BERT and T5 were particularly influential. BERT, introduced by Google, brought a novel approach to understanding language by employing bidirectional training. Unlike GPT-2, which processes text sequentially from left to right, BERT reads text in both directions simultaneously. This bidirectional method allowed BERT to capture the full context of a word based on the words around it, significantly enhancing its ability to perform tasks like reading comprehension and context-aware prediction. BERT's training involved masking parts of the text and predicting the missing words, which aided in achieving a more profound comprehension of the structure and meaning of language (Devlin et al., 2019). T5, also developed by Google a year later, took a different approach by framing all language tasks as text-to-text problems. This means that whether the task is translation, summarization, or question answering, T5 treats it as a transformation from one text to another. This unified framework simplified the model's design and allowed it to be highly versatile across various tasks. T5 was trained on an extensive dataset and, in its largest version, included 11 billion parameters, demonstrating its ability to efficiently manage a diverse array of text-oriented tasks (Raffel et al., 2020).

Both BERT and T5 introduced methodologies that complemented and expanded upon the capabilities of GPT. While GPT-2 focused on scaling up its size and dataset to improve performance, BERT's bidirectional context and T5's text-to-text framework offered new ways to understand and generate language. These developments would spark the beginning of other organizations to create their own LLMs tailored for various applications. GPT-2, despite being one of the most impressive models at the time, suffered from the same issues as its predecessor. This included challenges in maintaining coherence over longer passages and generating highly contextually accurate responses (Komatsuzaki, 2020). Therefore, OpenAI continued the trend of scaling up its models and rapid development to introduce its new flagship model a year later, the GPT-3 (Brown et al., 2020).

2.2.3 What makes LLMs Large?

In 2020, GPT-3 advanced the concept of LLMs to a new level. It contained an unprecedented architecture comprising 175 billion parameters and 96 layers, which made it a hundred times larger than GPT-2 and vastly outperformed its predecessors. This remarkable scale not only heightened the model's capacity for understanding and generating human-like text but also highlighted the inherent advantages of increasing model complexity and training data volume. This observation raises important questions regarding the reasons behind the substantial improvements in the capabilities of LLMs as more parameters are introduced, as well as how these large number of parameters are integrated within the model's architecture, using the GPT-3 as an example (Brown et al., 2020).

As detailed in previous sections, the model's architecture consists of a vast network of interconnected components, precisely the parameters or weights the model learns during its training process. For GPT-3, the training data this time consisted of the entire common web, all of Wikipedia, and several book collections. GPT-3 has a total of 175 181 291 520 weights, separated into 27 938 matrices falling under seven categories. These categories represent the various types of matrices involved in processing the data through the model's layers. Essentially, all operations performed in an LLM are matrix multiplications, each serving a specific purpose. As elaborated in chapters 2.1.3 to 2.1.5, the transformer architecture consists of word embeddings, the attention mechanism, and feed-forward networks, which collectively constitute all seven layers, namely the bespoke Key, Query and Value weights, as well as the weights of the embedding and unembedding layer and the projection matrices. The embedding layer converts words or tokens from the model's vocabulary into numerical vectors. GPT-3 has a vocabulary size of 50 257 tokens, and each token is represented by a vector with 12 228 dimensions. This means the embedding layer contains about 617 million weights. These weights start randomly but are adjusted during training to capture the relationships between different tokens better (Brown et al., 2020).

Conversely, the unembedding layer maps these numerical vectors back to the original vocabulary size of 50 257 tokens. Like the embedding layer, this layer also has around 617 million weights, which are refined during training to improve the accuracy of the generated text. The Key, Query, and Value layers all consist of matrices with 12 288 columns and 128 rows. Each layer then amounts to 1,5 million weights, except the value layer, which accounts for around 3 million weights, due to essentially performing the operation twice. Subsequently, it results in around 6 million weights per attention head. Since the transformer architecture operates with the previously elaborated multi-head attention, each block contains 96 heads of attention, which results in around 600 million parameters. Since the transformer iterates through these blocks multiple times, 96 layers in the case of GPT-3, the total number of parameters for the attention mechanism is around 58 billion. The rest of the parameters are in the feed-forward neural networks between the different attention blocks, called the up-projection and down-projection layers. The up-projection layer takes the input from the previous layer, which has 12 288 dimensions and projects it to a higher-dimensional space of 49 152. This enables the model to identify more intricate patterns and relationships in the data. The down-projection layer then maps this higher-dimensional representation of 49 152 back to the original size of 12 288 dimensions. This step compresses the information back into a manageable size while retaining the complexity identified in the higher-dimensional space. These layers are also repeated over the 96 layers, which combined results in the 115 billion parameters (Brown et al., 2020).

Tab. 1: Calculation of the total number of parameters of GPT-3

Parameters of GPT-3	**Rows**	**Columns**	**Heads**	**Layers**	**Parameters**
Embedding	12 288	50 257			617 558 016
Key	128	12 288	96	96	14 495 514 624
Query	128	12 288	96	96	14 495 514 624
Value & Output	128	12 288	96	96	28 991 029 248
Up-projection	49 152	12 288		96	57 982 058 496

Down-projection	12 288	49 152		96	57 982 058 496
Unembedding	50 257	12 288			617 558 016
Total Parameters					**175 181 291 520**

Source: Adopted from: (Brown et al., 2020)

An increase in the number of weights enables the model to capture and retain finer distinctions within the training data, thereby enhancing its ability to produce more precise and sophisticated responses. The enormous number of weights allows GPT-3 to capture a vast amount of information and subtle nuances in language. This design allows the model to manage various tasks, including responding to inquiries and creating content that is both coherent and contextually appropriate. The more weights the model has, the better it can learn from large datasets and produce high-quality, human-like responses (Kaplan et al., 2020). A larger model can also handle a bigger context window, meaning it can remember and consider more text at once. This enables it to comprehend and produce replies based on a larger amount of information. The context window for GPT-3 was 1000 words or 2048 tokens (Jurafsky & Martin, 2024, p. 238). GPT-3 also marked the first demonstration of the drawbacks associated with LLMs. The sheer size presents several challenges, particularly computational resources, cost, and environmental impact. Training, running, and storing LLMs require immense computational power and substantial memory resources (Bender, 2021). On the user side, the complexities and limitations of interacting with GPT-3 became apparent. Users faced difficulties with response relevance and coherence, and there were challenges in steering the model to meet specific needs or maintain consistent, contextually appropriate interactions. To address some of these limitations, instead of further increasing the scale of the LLM, OpenAI opted for a fine-tuning approach. This decision aimed to enhance the model's ability to adhere more closely to human instructions when provided with a prompt. The resulting model placed greater importance on the nuances of phrasing and representation in its generated responses. This iteration was referred to as InstructGPT (Ouyang et al., 2011).

2.2.4 An Incremental Update – ChatGPT

From a user's perspective, interacting with GPT-3 involves crafting prompts to elicit useful responses. The generated response would be heavily influenced by the user's prompt, which gave rise to a field called "prompt engineering." Prompt engineering is the practice of carefully designing and crafting prompts to get the best possible responses from an LM. It involves figuring out how to phrase questions or instructions in a way that guides the model to produce more accurate or practical results (Chen et al., 2023). In addition to the user prompt, the model is given several types of contexts to generate a response. This entails the conversation history, where the model retains the ongoing conversation's history to maintain coherence and relevance in its responses, the data it was trained on, and the system instructions, which are guidelines that shape how the model should respond. The response can then be further influenced by adjusting its hyperparameters. One of the most notable is temperature, a parameter controlling the randomness of the model's output. A lower temperature (e.g., 0.2) makes the model's responses more deterministic and focused, while a higher temperature (e.g., 1.0) introduces more randomness and variety to the responses. Once the prompt is submitted, GPT-3 processes it and generates a response based on all the factors mentioned above. This process is called inference. During inference, the model applies its learned patterns and knowledge to produce relevant outputs in response to the user's input. It is also during this step that GPT-3 showcased the ability to perform in-context learning, a capability of LMs where they learn to perform a task by seeing examples within the same input. Instead of needing explicit training, the model can understand and apply patterns based on the examples provided in the prompt (Brown et al., 2020).

Despite the impressive capabilities of GPT-3, the model had to be carefully tweaked and prompt-engineered to get the desired results. This process could be time-consuming and required significant trial and error to ensure the model produced relevant and accurate responses. The human feedback regarding the limitations of GPT-3 led to OpenAI fine-tuning the model, specifically for the context of human interaction. The paper by OpenAI in January 2022 tweaked GPT-3 to become InstructGPT by training it specifically to follow instructions better. They did this using a large dataset of prompts paired with high-quality, human-generated responses. This helped the model learn to understand and respond more accurately to detailed instructions. This adjustment made it easier for users to get the desired results with less effort in crafting prompts. Instead of just generating text based on prompts, it was designed to understand and act on detailed instructions more effectively. This enhancement aimed to improve the overall interaction quality of the model (Ouyang et al., 2011).

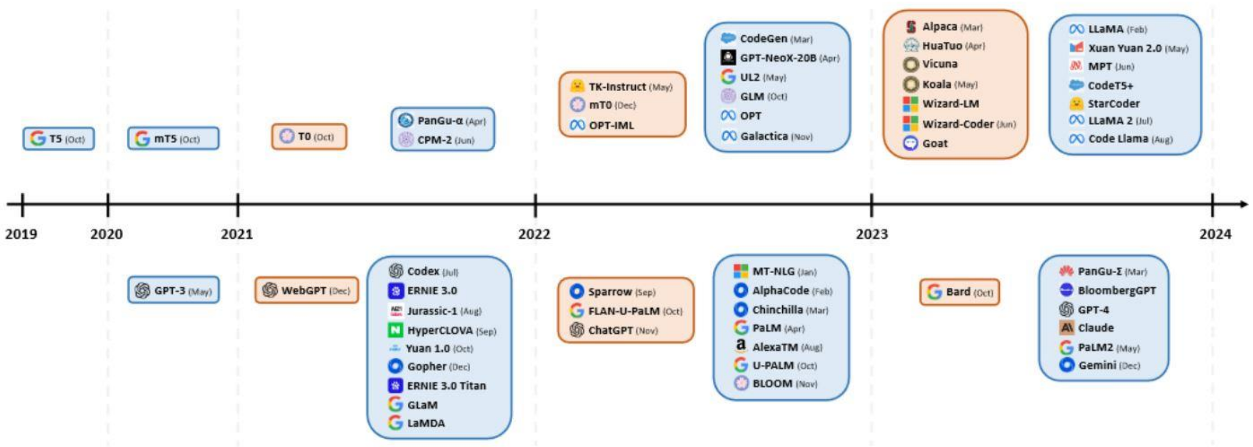
InstructGPT laid the foundation for what would then go on to become ChatGPT, also known as GPT-3.5. Launched in November 2022, GPT-3.5 marked a breakthrough moment, as it was the first generative text model to achieve widespread acceptance and utilization among the general public (Carr, 2023). ChatGPT not only improved upon InstructGPT in terms of conversational abilities but also streamlined the overall process. Its improved interface and ease of use made it more accessible and intuitive for everyday users. The success of ChatGPT has fundamentally transformed how the world perceives AI and the field of NLP as a whole, being one of the greatest advancements in technology to date. Companies and individuals began to explore the myriad applications of ChatGPT, ranging from customer service automation to content generation, tutoring in various subjects, and even creative storytelling. With the rising prominence of ChatGPT, numerous organizations made substantial investments in incorporating LLMs into their operations. Due to this widespread success, it was not long before the introduction of the next generation of LLMs (Ray, 2023).

2.2.5 The Current State of LLMs

After ChatGPT, the world experienced a significant shift in AI as the power and potential of these models became evident across various sectors. What began as a niche technology for NLP soon evolved into a cornerstone of modern AI, driving innovations in many industries and beyond (Ray, 2023). The current and fourth generation of LLMs introduced with GPT-4, released in 2023, represents a continuation of trends that have been developing for years. The foundations laid by early LMs have been built upon, resulting in systems that are not only more widespread but also more refined. These advancements have led to greater accuracy, versatility, and integration into everyday applications, making LLMs indispensable tools for businesses, researchers, and individuals (Achiam et al., 2023). GPT-4 further advanced these capabilities by scaling up again. While specific details about the number of parameters have not been disclosed, estimates claim that GPT-4 has 1,76 trillion parameters. GPT-4 brought several significant improvements over its predecessor. It enhanced the ability to understand and generate more nuanced and contextually accurate responses, making conversations feel more natural and coherent (Patel & Wong, 2023).

Additionally, GPT-4 and GPT-4o showed better handling of complex queries, offering more precise and thoughtful answers. They also introduced the ability to process and reason with larger amounts of information due to increases in the context window up to 128 000 tokens or about 96 000 words, leading to better outputs (OpenAI, 2024). Various organizations have also focused on LLM development following these events. Google, already a key player in the AI space, expanded its efforts by developing advanced models and refining its existing technologies. This led to its current model, Gemini 1.5 (Georgiev et al., 2024). Similarly, companies like Meta contributed their innovations to the space and open-source community by introducing the Llama model (Touvron et al., 2023). This surge in activity led to a diverse array of LLMs, each tailored with unique architectures and specialized for various applications. Some organizations have even focused on creating domain-specific models for particular fields or use-cases. In contrast, others have pursued generalized solutions with broad applicability, expanding to different domains outside of NLP. This proliferation of models reflects the increasing recognition of LLMs' potential and the growing demand for their integration into specific use cases. The competition and collaboration in this space have driven rapid advancements, effectively evolving into a race of which organization can produce the largest and most competitive model, as seen in Figure 8 (Naveed et al., 2024).

Figure 8: Timeline of LLM releases up to 2024



Note: Blue cards represent “pre-trained” and orange cards “instruction-tuned” models

Source: (Naveed et al., 2024)

Despite the diversity in models and methodologies, the core architecture across these systems remains fundamentally similar. Most contemporary LLMs are built on variations of the transformer architecture, which, while powerful, brings inherent challenges. As a result, despite the different approaches taken by organizations, the models share fundamental issues related to their design and implementation. These problems, already evident with the introduction of GPT-3, have been exacerbated even more as models have scaled up. With each new generation, as models increase in size and complexity, the problems tend to become more pronounced. The amplification of issues such as costs, computational resources, and environmental impact becomes more evident. Despite their enhanced capabilities, larger models struggle with the same underlying flaws at a grander scale. The most significant of these issues is the phenomenon of hallucinations (Patil & Gudivada, 2024).

2.3 Understanding and Tackling Hallucinations: Impact, Causes, and Solutions

2.3.1 Introduction to Hallucinations and their Impact

Hallucinations in the context of LLMs refer to the phenomenon where the outputs of a model sound plausible and coherent but are factually incorrect. The term “hallucination” is derived from a psychological context where individuals perceive things that are not present. Similarly, LLMs can produce outputs that sound plausible but are fabricated. Due to LLMs' fast rise in popularity, many different sectors have their share of problems resulting from hallucinations (Ji et al., 2022).

Numerous high-stakes industries, such as law, healthcare, and finance, face significant challenges in adopting LLMs. The primary concern lies in the robustness and reliability of the outputs generated by these models. In sectors like law, where the stakes often involve critical legal outcomes, inaccuracies or ambiguities in LLM-generated content could lead to profound consequences for individuals and organizations. Legal practitioners rely on precision and nuanced interpretations that current LLMs may struggle to provide (Dahl et al., 2024). Similarly, in healthcare, incorrect recommendations or misunderstandings of patient data can endanger patient safety and treatment efficacy. The medical field demands a high level of accuracy that existing LLMs have yet to consistently achieve (Ahmad et al., 2023). The financial sector is not exempt from these risks; faulty data analysis or flawed algorithmic trading strategies can result in substantial financial losses. Given these potential repercussions, it becomes evident that implementing LLMs into such high-stakes environments would be equivalent to negligence (Kang & Liu, 2023).

Hallucinations also became a problem in non-critical industries. Many "AI-powered" products have been brought into the market to improve the efficiency of various business operations. These AI tools are designed to streamline processes such as data analysis, customer service, and marketing automation. However, when these systems produce faulty outputs, it can lead to misguided decisions (Ghose, 2024). For instance, an AI employed in market research may misinterpret consumer data and generate unreliable insights, ultimately skewing business plans. Additionally, in customer service applications, hallucinations can result in inappropriate responses being delivered to clients, potentially damaging brand reputation and customer trust (Taplin, 2024).

This is also essentially the tech industry's fault for not emphasizing the robustness and accuracy of AI systems before releasing them into the marketplace. Tech Companies often rush to adopt AI technology, aiming to stay competitive or capitalize on trends, sometimes overlooking thorough testing and validation processes that would ensure the reliability of these tools. Consequently, numerous live demonstrations, such as Google's Bard and Microsoft Copilot chatbots, failed in this aspect, highlighting that LLMs are not yet ready to replace established technologies like search engines (Wong, 2023).

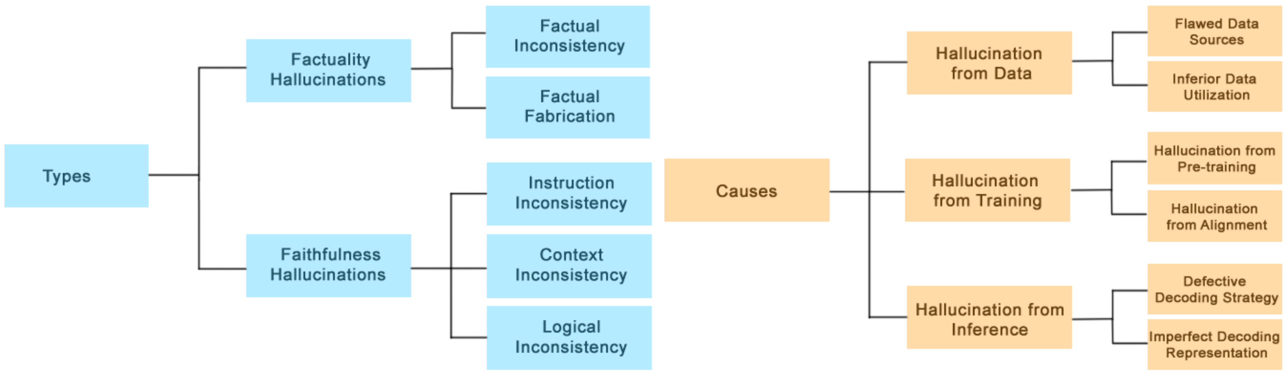
Arguably, the biggest negative impact of hallucinations is, however, in the realm of news and social media. As AI-generated content becomes increasingly prevalent, the potential for misinformation or "fake news" rises dramatically. The already existing lack of fact-checking, combined with the rapid sharing of incorrect information, leads to widespread misinformation, resulting in irreparable reputation damage in the worst cases (Rhys Leahy et al., 2021). There have even been instances where hallucinated LLM content was reported in the mainstream news. This was often due to a lack of fact-checking processes within media organizations. In a time when speed frequently takes precedence over accuracy, journalists are sometimes pressured to publish content quickly without verifying its authenticity (Huet, 2024). Consequently, such lapses can damage news outlets' credibility and perpetuate false narratives that influence public perception and opinion. As more people rely on social media platforms as primary news sources, the consequences become even more pronounced, and the general distrust of information on the internet rises (Yadlin & Marciano, 2024).

The occurrence of hallucinations is a notable challenge across various LLMs, highlighting limitations inherent to their design. This issue is not only frequent, with hallucination rates being up to 27%, but also complex (Metz, 2023). It affects the outputs across different applications and domains and has different causes and varying types of hallucinations. Therefore, and due to the aforementioned reasons, addressing and mitigating hallucinations remains an ongoing field of research. This also begs the question of how and why hallucinations occur in the first place (Huang et al., 2023).

2.3.2 Why do Hallucinations Occur?

Hallucinations in LLMs occur due to several factors related to their design and functionality. Fundamentally, LLMs operate as black boxes, meaning their internal decision-making processes are not readily interpretable or transparent. This obscurity complicates the understanding of how these models generate responses and where these problems originate. Nonetheless, extensive research has been conducted to identify shared underlying factors and to classify the various types of hallucinations. In general, there exists a significant lack of uniformity in the classification of hallucinations, resulting in the absence of a standardized system. Consequently, the taxonomy presented here is derived from sources that offer a comprehensive examination of the subject, compared with common classifications (Maleki et al., 2024).

Fig. 9: Classification of Types and Causes of Hallucinations in LLMs


Types
Factuality Hallucinations
Factual Inconsistency
Factual Fabrication
Faithfulness Hallucinations
Instruction Inconsistency
Context Inconsistency
Logical Inconsistency
Causes
Hallucination from Data
Flawed Data Sources
Inferior Data Utilization
Hallucination from Training
Hallucination from Pre-training
Hallucination from Alignment
Hallucination from Inference
Defective Decoding Strategy
Imperfect Decoding Representation

Source: Adopted from: (Huang et al., 2023)

Hallucinations can broadly be categorized into two main types: factuality hallucinations and faithfulness hallucinations. Factuality hallucinations refer to instances where a model generates information that is factually incorrect or unverifiable. This occurs when the model fabricates details like names, dates, or events. Conversely, faithfulness hallucinations relate to discrepancies between the generated output and the original prompt. In these cases, while the information may not be factually incorrect, it fails to reflect the prompt's intentions accurately. These two categories can then further be classified into sub-categories. Factuality hallucinations are categorized into factual inconsistencies and factual fabrications. Factual inconsistencies occur when the generated content contains contradictions or conflicts with established knowledge. For example, a model might state that a historical figure was born in two different years within the exact text (Huang et al., 2023).

On the other hand, factual fabrications involve the complete invention of information that does not exist, such as creating fictitious events that seem plausible but are made up. Faithfulness hallucinations can be broken down into the categories: instruction inconsistency, context inconsistency, and logical inconsistency. Instruction inconsistency arises when the output diverges from particular instructions given in the prompt. For instance, if a user requests a summary of a text but receives an analysis instead, the response may fail to align with what was explicitly asked for. Context inconsistency occurs when the generated output does not appropriately consider previous information provided within the conversation or text. This could manifest as contradictions between earlier statements and subsequent outputs disregarding prior context, leading to confusion about what has been established. Logical inconsistency is when the reasoning or conclusions presented in the output do not follow logically from its premises. For example, if a model provides reasoning based on flawed assumptions, it creates an impression of coherence while lacking actual logic (Huang et al., 2023).

Moreover, hallucinations can be attributed to three primary causes: Hallucinations from data, from training, and from inference. Pre-training constitutes a fundamental component of the LLM development process, serving as the foundation for the model's capabilities and knowledge base. However, this phase is also a key contributing reason to occurrences of hallucination, primarily attributable to two key factors: the training data employed and the training process itself (Y. Zhou et al., 2024). Hallucinations from data in LLMs can be traced back to flawed data sources and inferior data utilization. Flawed data sources can encompass misinformation and biases and the boundaries of knowledge within the dataset. Misinformation leads the model to learn and perpetuate inaccuracies, while biases skew the model's outputs towards certain viewpoints, often reflecting societal prejudices embedded in the data (Weidinger et al., 2021). Knowledge boundaries refer to the gaps in the data where the model lacks exposure to specific topics or facts, leading it to generate content based on incomplete or incorrect assumptions (Lee et al., 2023). Inferior data utilization refers to the model's difficulty in accurately recalling, interpreting, and applying the knowledge stored within its parameters. One of these issues is knowledge shortcuts, where the LLM relies on superficial patterns in the data, such as the frequency of words appearing together, rather than truly understanding the underlying concepts. Similarly, knowledge recall failures include long-tail knowledge failures, where the model struggles with obscure facts, and complex reasoning failures, where it fails to accurately perform multi-step reasoning or connect information (Bender, 2021).

Hallucinations from training can be categorized into those arising from pre-training and those from alignment or fine-tuning. Hallucinations from pre-training relate to fundamental issues in how the model is initially trained. These can include architecture flaws, where the model’s design or structure limits its ability to capture and process knowledge accurately, and suboptimal training objectives, where the goals set during training do not align well with producing accurate or reliable outputs for the subsequent utilization (C. Wang & Sennrich, 2020). On the other hand, hallucinations from alignment involve issues that emerge when adapting the model for specific tasks or aligning it with user expectations. These include capability misalignment, where the model’s trained abilities do not match the requirements of its intended application, and belief misalignment, where the model’s generated outputs reflect incorrect or misinformed assumptions (Gekhman et al., 2024).

Hallucinations from inference can be attributed to defects in the decoding strategy and imperfections in decoding representation. A defective decoding strategy refers to issues in generating outputs from the model's internal representations, such as inherent sampling randomness. This randomness can cause the model to produce variable and sometimes inaccurate responses, as the sampling process may not consistently align with the most relevant or accurate information (Holtzman, Buys, Du Li, Forbes, & Choi, 2020). Imperfect decoding representation involves limitations in how the model's output is constructed based on its learned knowledge. This can include insufficient context attention, where the model fails to fully incorporate all relevant information from the input, and the softmax bottleneck, where the model's probability distribution over possible outputs constrains its ability to generate diverse or accurate responses (Chang & McCallum, 2022; Miao et al., 2021).

2.3.3 How to Detect and Mitigate Hallucinations

Together, these three factors contribute to the model's tendency to produce hallucinated outputs. To address this challenge, researchers and developers have focused not only on the classification and causes of hallucinations but also on the implementation of hallucination detection, as well as various hallucination mitigation strategies. Since there are two main categories of hallucination, namely factuality and faithfulness hallucinations, hallucination detection revolves primarily around those two types (Tonmoy et al., 2024). Factuality hallucination detection involves identifying instances where the model generates information that is inaccurate or potentially deceptive. Therefore, a common technique for factuality detection involves fact-checking the model's output against external knowledge bases or other verified databases (J. Li et al., 2024). On the other hand, faithfulness hallucination detection assesses whether the model's output faithfully represents the input data or prompt. For faithfulness detection, techniques involve analyzing the alignment of the model's attention mechanisms with the input text. This approach checks whether the parts of the input that the model focuses on during generation are relevant and accurately reflected in the output (Maynez et al., 2020).

Based on the previous elaboration of the causes of hallucinations, mitigation strategies consist of the same categories, namely, mitigation of data-related hallucinations, training-related hallucinations, and inference-related hallucinations. Again, each category consists of various sub-categories of mitigation strategies, which revolve around addressing the different sub-categories of hallucination causes. Overall, the causes of hallucinations and their corresponding mitigation strategies can broadly be classified into two main categories: model development and inference (Tonmoy et al., 2024). The development of models is fundamentally focused on resolving the challenges associated with data quality and the training process that led to hallucinations. Mitigation strategies for data include approaches such as reducing misinformation and biases through factual data enhancement and debiasing techniques. For instance, this entails improving the quality of training datasets to prevent the incorporation of misleading information and biases. This might involve curating datasets more carefully, implementing factuality checks, and ensuring that the data provides a balanced and accurate portrayal of information (Gunasekar et al., 2023).

Strategies addressing the training-related hallucination causes are, for example, mitigating pre-training-induced hallucinations via architecture improvements or pre-training objective improvements. An architectural improvement is, for example, refining how the model processes and stores information to reduce the likelihood of generating hallucinated outputs (Z. Li et al., 2023). Pre-training objective improvements mitigate hallucinations via better alignment with factual accuracy or incorporating external verification mechanisms (Lee et al., 2023).

Fundamentally, the majority of hallucinations in LLMs arise primarily from issues stemming from the internal training data and the model's learned representations rather than from the inference process (Huang et al., 2023). However, addressing these issues is complicated by the fact that many of the most powerful LLMs today are closed-source. This means that the underlying data, model architecture, and training processes are restricted and inaccessible to external researchers. As a result, the progress in mitigating hallucinations heavily depends on the efforts and priorities of the organizations that develop these models. However, for external researchers and users, the inference stage remains a crucial tool to manage and reduce hallucinations (Gairola, 2024).

The inference stage also offers several strategies to mitigate hallucinations by refining how the model generates and evaluates its outputs in real-time. Overall, there are three main approaches. The first is retrieval-augmented generation (RAG), where the model supplements its internal knowledge with relevant information retrieved from external databases. This helps ensure that responses are grounded in accurate, up-to-date facts rather than relying solely on the model's learned patterns (Kang, Ni, & Yao, 2024). The next central technique involves self-refinement through feedback and reasoning, where the model iteratively improves its outputs by evaluating and refining them based on additional reasoning or feedback mechanisms. This allows the model to self-correct any inconsistencies or potential errors in its initial response (Madaan et al., 2023). Finally, there is prompt tuning, a process that fine-tunes a model by learning optimal prompt embeddings through training rather than manually refining the input prompts. By adjusting these learnable parameters, prompt tuning enhances the model's capacity to produce more precise and relevant responses, reducing the likelihood of hallucinations and ensuring better alignment with the intended query (Lester et al., 2021).

To conclude, despite hallucination being one major issue among many others in LLMs, it can also be dissected into various sub-components, each with distinct causes and characteristics. Because of the multitude of different causes and types of hallucinations, mitigation strategies must be precisely tailored to address the specific issues involved. This precision means that strategies designed to counteract one type of hallucination may involve vastly different approaches from those targeting another. However, both can lead to significant improvements in the model’s output. The result is a complex permutation of possibilities, where the effectiveness of a given strategy is highly dependent on the type of hallucination it addresses and the specific cause it seeks to combat. For example, a strategy focused on improving data quality might reduce hallucinations related to factual inaccuracy. At the same time, a method aimed at refining the model’s understanding of context might mitigate issues related to coherence. Despite these varying approaches, each strategy contributes to the overall reduction of hallucinations, highlighting the intricate and multifaceted nature of addressing this challenge (Huang et al., 2023).

3. Research Methodology

The upcoming section of this thesis discusses the experiment designed to explore the potential of ensemble learning methods to mitigate hallucinations and enhance the performance of LLMs. Based on the previous chapter about hallucinations, this experiment specifically targets factuality hallucinations using the inference stage of the model as a mitigation strategy. Ensemble methods fall under the category of self-refinement through feedback and reasoning. This technique allows the model to iteratively evaluate and enhance its responses through feedback, thereby reducing hallucinations. The output of the ensemble method is then fact-checked against an external knowledge base as a detection strategy using NER. The experiment is structured as follows: the first sub-chapter details the experimental setup, and the second outlines the procedure.

3.1 Experiment Setup

3.1.1 Utilizing NER to Detect and Evaluate Factuality Hallucinations

NER is an essential technique in NLP that focuses on identifying and classifying named entities within a text into predetermined categories such as people, places, organizations, dates, and more. By analyzing text through NER, one can extract specific entities integral to the meaning of the content. The process typically involves tokenization, followed by classification, where each token is labeled according to the entity it represents or if it represents no entity at all (Pakhale, 2023). In the context of factuality hallucination detection for LLMs, NER serves as an external knowledge base for evaluating the accuracy of the information generated by the model. Specifically, it is used to compare the named entities in the model's output with those in a verified dataset (S. Wang et al., 2023).

The DocRED dataset is employed for this experiment, a large-scale dataset designed for document-level relation extraction, which includes human annotations of named entities and their relationships. Using NER, the entities mentioned in the model's output can be systematically compared with those present in the DocRED dataset. Suppose the model generates entities that do not align with those in the dataset, either by introducing entities that do not exist in the source data or by misidentifying them. In that case, this discrepancy indicates a factuality hallucination. Through this method, NER helps detect factual inaccuracies and provides a measurable way to assess the model's performance in generating factually correct outputs. The results can be quantified by calculating precision, recall, and F1-score, which assess how accurately the model identifies and classifies these entities over multiple predictions (Yao et al., 2019).

3.1.2 Ensemble Methods as Mitigation Strategy

Ensemble methods are a class of ML techniques that combine multiple predictions of models to improve overall performance and robustness. By aggregating the outputs, ensemble methods aim to reduce errors and variability (Z.-H. Zhou, 2012, pp. 15–17). In the context of hallucination mitigation, ensemble methods fall under the broader category of "self-refinement through feedback and reasoning," where the model iteratively evaluates and improves its outputs by incorporating multiple viewpoints or model outputs. Thereby reducing inconsistencies and errors, making them potentially useful as a mitigation strategy for hallucinations in LLMs (Pham & Vo, 2024). Ensemble methods can be classified into various subcategories. However, no uniform agreement exists on the exact number of these categories, ranging from two to over five, depending on the source. Nonetheless, the predominant sub-categories consistently recognized in all classifications are bagging and boosting. These two techniques will serve as the primary focus for the subsequent experiment (Dasari et al., 2023).

Bagging and Boosting are the two most common and oldest types of ensemble techniques, both of which offer a distinct approach to improving the model output. In essence, bagging and boosting can be applied to any methodology that produces an output and improves on it. Bagging, also known as "Bootstrap Aggregating," does this in a parallel manner. This would typically involve training multiple independent models on different subsets of the data and then averaging their outputs. This approach helps reduce outliers and increases the stability of the model's predictions (Dasari et al., 2023). LLMs, however, being stochastic in nature, produce different outputs even when given the same prompt (Jurafsky & Martin, 2024, pp. 231–234). Transferring the concept of bagging for the use case of hallucination mitigation in LLMs means generating multiple outputs from the same model and selecting the final response through a majority vote. This adaptation seeks to limit the influence of hallucinations from individual outputs. Therefore, the bagging part of the experiment aims to prove if aggregating the predictions reduces the overall occurrence of hallucinations in the final result.

Boosting, on the other hand, takes a more sequential approach. It sequentially trains multiple models, each focused on correcting the errors made by its predecessor. Unlike bagging, where models are trained independently, boosting builds models in a sequence, with each new model giving more weight to the data points that were misclassified by previous models. The final prediction is a weighted aggregation of all the models' outputs, allowing boosting to create a robust overall prediction from several weaker ones (Freund & Schapire, 1997). Transferring the concept of boosting to LLMs to mitigate hallucinations involves identifying problematic responses and refining them over multiple iterations within the same model to correct factuality hallucinations. This approach aims to correct hallucinations that could occur by using feedback mechanisms to correct their errors. Each iteration focuses on addressing the factual inaccuracies detected in previous outputs if there are inaccuracies. Therefore, the experiment part of boosting seeks to prove if the model can progressively improve its accuracy and reduce hallucinations over multiple outputs.

3.1.3 Selecting a Suitable Model

Selecting the right LLM for the experiment must be carefully considered, given the vast array of options available with over 100 different models currently on the market. Determining the "best" model is not straightforward, given that there are so many choices. Performance is influenced by various factors beyond just the number of parameters. Each LLM offers its own unique set of advantages and trade-offs (Maslej et al., 2024). Therefore, multiple criteria were considered when selecting a suitable model for the experiment. The three main criteria were recency of release, cost efficiency, and user adoption. The recency of release is significant because newer models are typically built on larger datasets and incorporate the latest advancements in AI architecture. They tend to offer improved performance and better accuracy, outperforming previous iterations. Cost efficiency was especially important for the experiment because LLMs can quickly become expensive, especially when processing larger datasets. A cost-effective model can conduct the experiment multiple times, and the dataset can be more extensive. Adoption by users was an essential criterion regarding the results of the experiment because it reflects the reality of model usage. The best-performing technology does not necessarily have to be the status quo. That means that even if a model performs well in the experiment, the value diminishes if it is not really used.

The first key factor for exclusion was the range of each LLM. Only those considered general purpose were included, while those tailored to specific domains or use cases were discarded. Similarly, only the latest and most advanced versions of these LLMs were taken into account, leaving out any earlier versions. The following selection criterion is the user base or popularity of the models. This further narrows down the models to only a handful of the most recent models of the largest organizations, namely GPT, Gemini, Llama, Claude, and Mistral (Bommasani et al., 2023).

Which then further boils down to the question of cost efficiency. In essence, there are two main price models: pay-by-token and hosting. Open-source LLMs such as Llama and Mistral mainly represent the hosting category, while most closed-source LLMs, like GPT, Gemini, and Claude, are pay-by-token. Despite common beliefs, one solution is not necessarily cheaper than the other. Hosting models, for example, involve running the model on the in-house infrastructure or cloud, which allows for more control and flexibility. The costs are derived from the computing resources required to run these models and potential licensing fees. In addition, hosted models also have their downsides, mostly related to the additional complexity of setting up and running the model. The costs of the pay-by-token price model are derived from paying for the number of tokens for input and output. How much is paid for a specific number of tokens is determined mainly by the strength of the model. Hidden variables like poor configuration and optimization can additionally increase costs. Therefore, choosing and properly configuring a suitable model is vital when using pay-by-token LLMs. These models also have the added benefit of their simplicity. Compared to hosting, pay-by-token LLMs act as Software-as-a-Service (SaaS), which enables quick and easy use via API calls. API calls are not without their own issues but are negligible in the case of a smaller-scale experiment. For these reasons, the models of choice are closed-source LLMs (Irugalbandara et al., 2024).

Based on these factors, in addition to the last criteria of model popularity, the most suitable choice is GPT. The current model, GPT-4 omni, also called GPT-4o, overall outperforms the other models, considering the aforementioned criteria. This model also has a variant, which, in addition to the mentioned criteria, is much more cost-effective. Therefore, the following experiment is conducted utilizing GPT-4o-mini as the model (R. Islam & Moushi, 2024).

3.2 Experiment Procedure

This section describes the experimental procedure, which involves the application of three different methods: standard NER, boosting NER, and bagging NER. While each method varies in its approach, they all follow the same general pipeline consisting of four key steps.

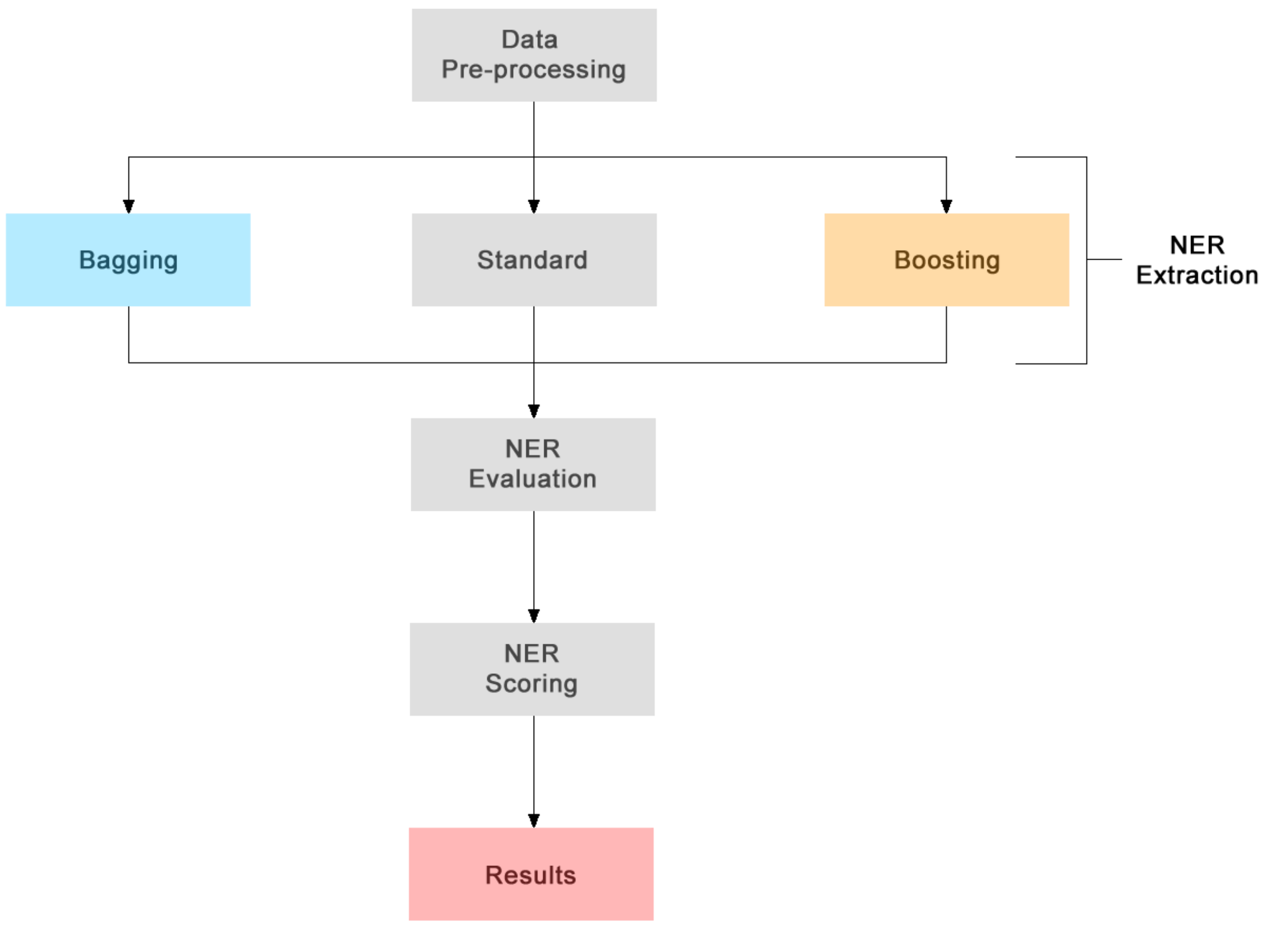
Data pre-processing: This step transforms the data of the DocRED dataset into clean and structured data, removing any irrelevant or unwanted information.

NER extraction: Once pre-processing is complete, the cleaned data is used by GPT-4o-mini to extract named entities.

NER evaluation: The extracted entities are then compared with the original dataset to determine their correctness.

NER scoring: The evaluation results are used to calculate quantitative metrics that assess the overall performance of the approach.

Fig. 10: Flowchart of the NER experiment



Source: Own Results

3.2.1 Data Pre-processing

Data pre-processing plays a pivotal role in enhancing quality, accuracy, and efficiency by ensuring that the input data is well-structured and free from unnecessary noise. In this experiment, the aforementioned DocRED dataset is utilized. The dataset is organized in JSON format and includes human-generated and AI-labeled annotations. The AI-labeled part of the dataset exceeds 100 000 entries but is generally lower in quality than the human-generated portion, which contains only around 3000 entries. For the sake of maintaining higher quality and reliability in the experiment, solely the human-annotated portion of the dataset was chosen. This human-verified data is generally more accurate than AI-generated labels, which can sometimes introduce mistakes. In addition, due to the nature of the experiment, only a smaller subsample of data is needed. Therefore, only a subset of 500 rows for further processing were selected, ensuring a manageable yet representative sample size.

The pre-processing of this dataset involved several steps aimed at refining and preparing the data. Initially, the text data was stored as lists of sentences, each broken down into smaller components. These lists were then reconstructed to form coherent documents, restoring the original structure needed for meaningful analysis. Each document in the dataset contains named entities, referred to as "vertexSet," alongside relationship data, known as "labels." However, since this experiment focused purely on NER and not relationship extraction, the relationship labels were deemed unnecessary and subsequently removed.

Furthermore, the entities in the vertexSet were disentangled into separate rows, with each row containing only the names of the entities. Additional information, such as the token position and sentence position, was removed. This transformation allowed for a cleaner, more organized dataset where only the essential entity names were kept for comparison purposes. Importantly, during this process, two specific entity categories, namely NUM (numerical data) and MISC (miscellaneous), were entirely excluded. The annotations for these categories were often arbitrary and lacked consistency. The end result of the pre-processing step was a refined dataset containing only the most relevant rows: the document title, the document text, and the named entities categorized into four key types: PER (person), LOC (location), ORG (organization), and TIME (dates and times).

3.2.2 NER Extraction

After pre-processing, the cleaned data is passed to the NER extraction module, along with the API key and the selected extraction mode. This module offers three distinct extraction methods: standard, boosting, and bagging. The chosen extraction mode determines how the NER process is handled. In standard mode, the process involves a straightforward API call without any additional layers of processing. However, in boosting and bagging modes, further refinements are applied

The initial mechanism is the same across all three methods. The document text, derived from the pre-processed data, is submitted to GPT-4o-mini as part of a user prompt. The corresponding system prompt instructs the model on what to do, specifically to extract named entities from the text.

The system prompt also defines how the extraction should be performed, ensuring consistency in the process and the output. Once the GPT-4o-mini generates a response, the extracted entities are further cleaned and formatted into a structured dataset for further evaluation. The enhanced extraction methods, boosting and bagging, take this process a step further. In boosting mode, the initial output from GPT-4o-mini is fed back into the model, along with the original text, but with an updated system prompt. This prompt changes the focus to correct specific mistakes. This iterative process acts as a correction mechanism, allowing the model to refine its initial output, and was repeated multiple times. By revisiting the original content with its previous output in mind, boosting increases the likelihood of capturing more entities. On the other hand, bagging follows a different strategy to improve entity extraction. Instead of refining a single output, bagging generates multiple outputs from GPT-4o-mini for the same document. These multiple versions are then evaluated to determine if the majority of the outputs, specifically more than half, contain the same entity. If an entity consistently appears across these generated versions, it is considered a reliable entity. This majority-vote approach in bagging tries to mitigate errors or inconsistencies that might occur in a single output, offering a form of robustness through redundancy.

3.2.3 NER Evaluation

After the extracted entities dataset has been created, the next step involves evaluating these entities against the corresponding data in the DocRED dataset. To make the comparison, the evaluation iterates through both datasets, row by row and column by column, examining the length and content of each list of extracted entities. Each list corresponds to a specific type of entity, such as "PER" for people, and is associated with a particular document from the dataset. Overall, this is a two-step evaluation process.

The first part of the evaluation compares the lengths of the lists from the extracted entities dataset to those in the DocRED dataset. This step helps identify potential errors based solely on the number of entities. There are three possible scenarios in this comparison:

1. The list of extracted entities is shorter than the list in the DocRED dataset. This means that certain entities have not been identified, leading to false negatives (FN). These missing entities are labeled as "Missing Entity."
2. The extracted entity list is longer than the corresponding list in the DocRED dataset. This indicates that extra, incorrect entities have been identified. These are classified as false positives (FP), labeled as "Redundant Entity."
3. If the two lists are the same length, there are no errors in terms of entity count, and the comparison moves to the next phase.

Following the length comparison, the actual content of the entity lists is evaluated. This step involves a fuzzy matching approach rather than strict matching. Typically, entities would only be considered correct if they align perfectly with the entries in the DocRED dataset. However, after experimentation, it was determined that an exact position matching is not feasible due to the inaccuracies of the generated output regarding token position. To address this, a fuzzy matching technique was chosen, allowing for some variation in the entity labels. In this approach, each extracted entity is compared to the corresponding entity in the DocRED dataset. In this comparison, it is evaluated if the ground truth list merely contains the entity. For this method to be effective, duplicates were also removed. If the entity is found within the comparison dataset, it is counted as a true positive (TP) and labeled as "True." On the other hand, if the entity is not found within the comparison dataset, the entity is marked as FP and labeled as a "Mismatched Entity." This matching strategy helps account for minor discrepancies regarding the entity position, which improves the robustness of the evaluation.

3.2.4 NER Scoring

The final module of the experiment focuses on calculating the quantitative metrics that measure the overall performance of the standard, boosting, and bagging approaches. This module takes the list of correctly identified entities and errors generated in the evaluation stage and uses them to compute performance indicators. Specifically, the scoring is based on three categories: TP (correctly identified entities), FP (incorrectly extracted entities that are not present in the ground truth and mismatched entities), and FN (entities in the dataset that were missed during extraction). True negatives (TN) refer to the correct non-classification of non-entities. This includes all words of a sentence that cannot be classified as entities. Due to them not being classifiable, there is no data in the DocRED dataset that can, therefore, not be compared. On the other hand, the focus is solely on how well the model identifies entities, which makes the other three metrics more significant. To assess the effectiveness of the NER approach, the module counts the total occurrences of TP, FP, and FN. These values are then used to calculate three core metrics:

1. Precision measures the proportion of correctly identified entities out of all the extracted entities. Higher precision indicates fewer FP.
2. Recall quantifies the proportion of actual entities that were successfully identified. A higher recall reflects fewer FN.
3. F1-score, the balanced mean of precision and recall, offers a fair assessment by combining both metrics into a single score. This metric is important to determine the method with the highest overall performance.

In this scenario, where there is only access to TP, FP, and FN, there is no option to calculate accuracy or specificity. This makes the precision, recall, and F1-score the most meaningful performance metrics to focus on. Alternative metrics exist, but in this case, they offer the same core concepts without much-added value. The confusion matrix is superior to other evaluation methods for this experiment because it directly captures classification outcomes without relying on underlying probabilities or model assumptions. This makes it a more straightforward and transparent approach to assessing model performance (Ting, 2011).

4. Research Findings

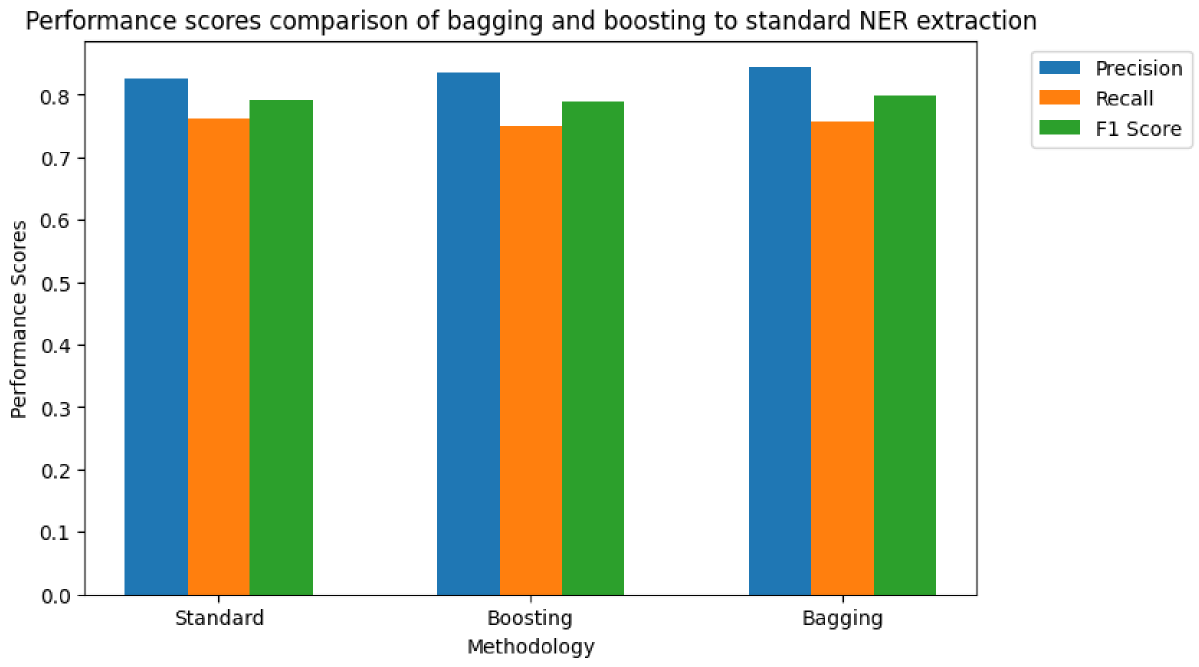
This chapter presents and analyzes the key findings of the experiment, providing a comprehensive overview of the results and their implications. The goal is to provide a well-rounded understanding of not only what the data reveals but also how these results hold up when considered alongside practical factors and potential limitations. The evaluation is divided into three sub-sections.

In the experiment results section, the raw results of the experiment are discussed. This section aims to give a clear and unbiased view of what the experiment achieved, focusing solely on the data produced. Next, the feasibility analysis takes a closer look at these results through a more practical lens. This section assesses the practicality of the results by weighing them against critical factors such as cost, effectiveness, and time. This puts the results in perspective by discussing whether the outcomes are sustainable in real-life implementation. Finally, the limitations section highlights the constraints and challenges faced during the experiment. This chapter outlines the factors that may have influenced the results and offers context for their interpretation.

4.1 Experiment Results

As previously elaborated, this section focuses on the direct outcome of the experiment. The most important result of the experiment is shown in Figure 11. **The tested methodologies of boosting and bagging lead to near-equal performance scores compared to the standard NER extraction without any additional correction mechanism.** The f1-scores, which determine the overall performance, are 79,9% in bagging, 79,2% in standard, and 78,9% in boosting. In the case of bagging, there was a minor overall improvement and even a slight decline in boosting. To further understand this result, it is necessary to dissect the performance scores and look at the different error types.

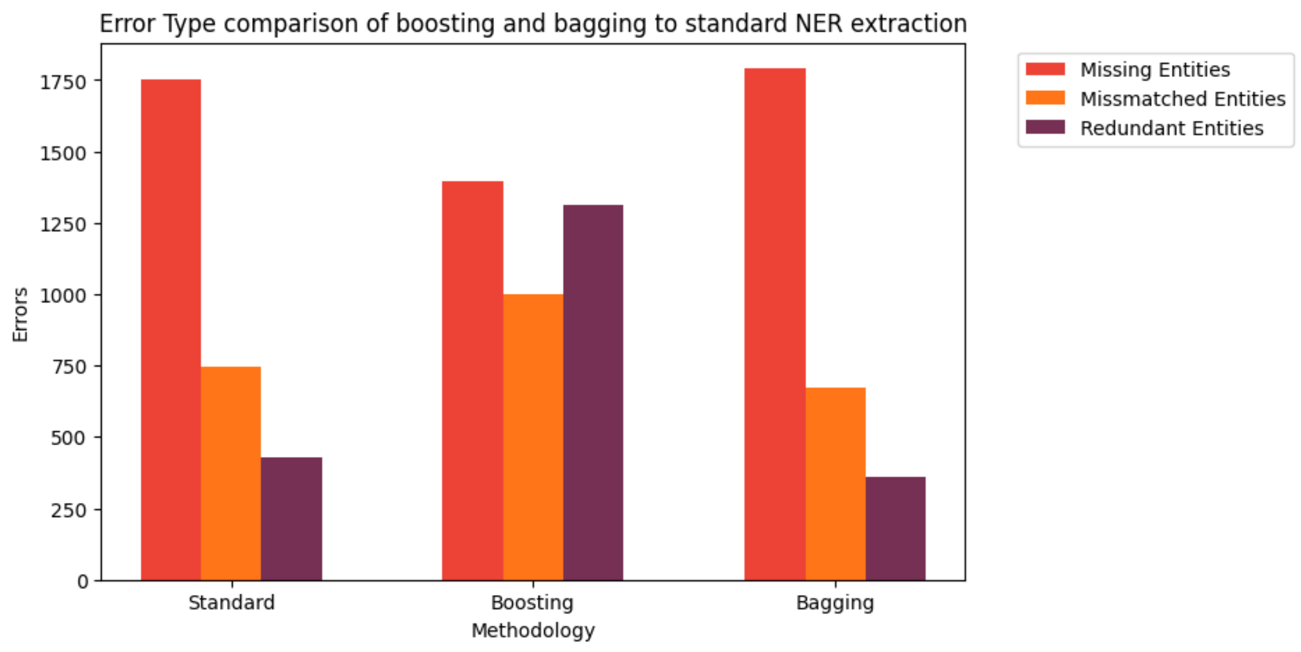
Fig. 11: Overview of the Final Performance Scores of standard, bagging and boosting



Source: Own Results

Further breaking down the FN and FP, from which the performance scores are evaluated, reveals why there are only slight changes, as shown in Figure 12. These are the three different error types that can occur during the evaluation, as discussed in the experiment procedure. **What becomes evident during the inspection is that changes do occur but are equalized or negligible during the final scoring section.** As seen in the figure, boosting led to the most reduction in missing entities (FN), while bagging led to a reduction in mismatched and redundant entities (FP).

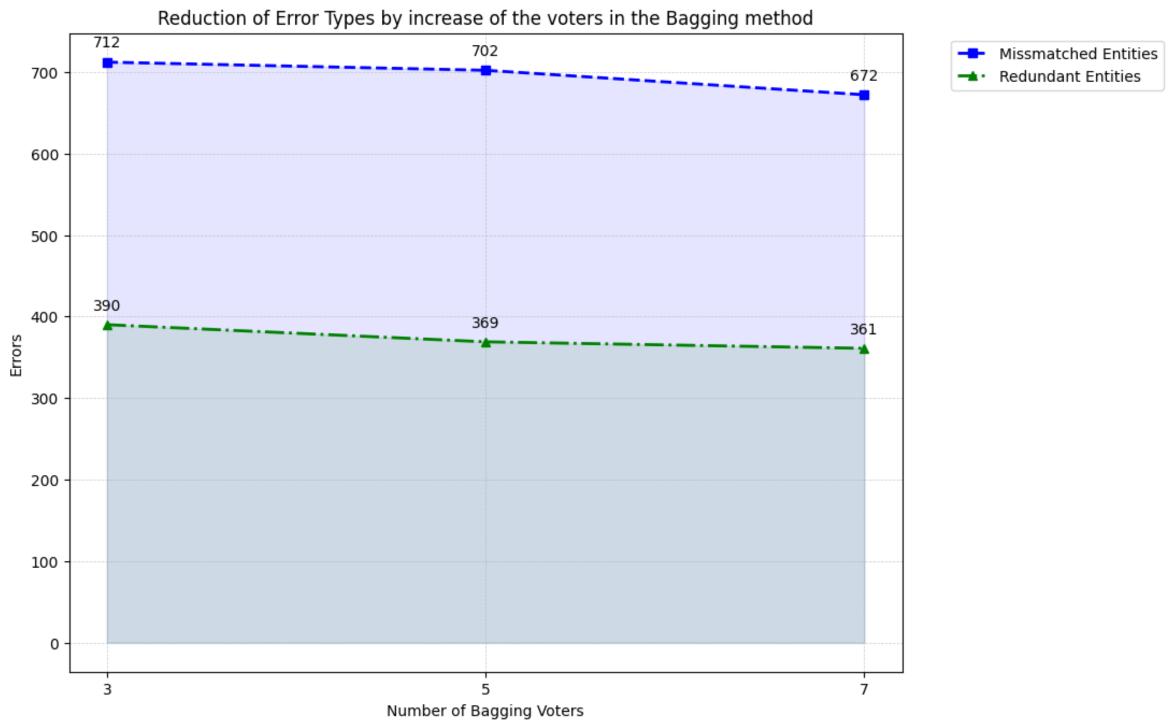
Fig. 12: Overview of the Error Type occurences of standard, bagging and boosting



Source: Own Results

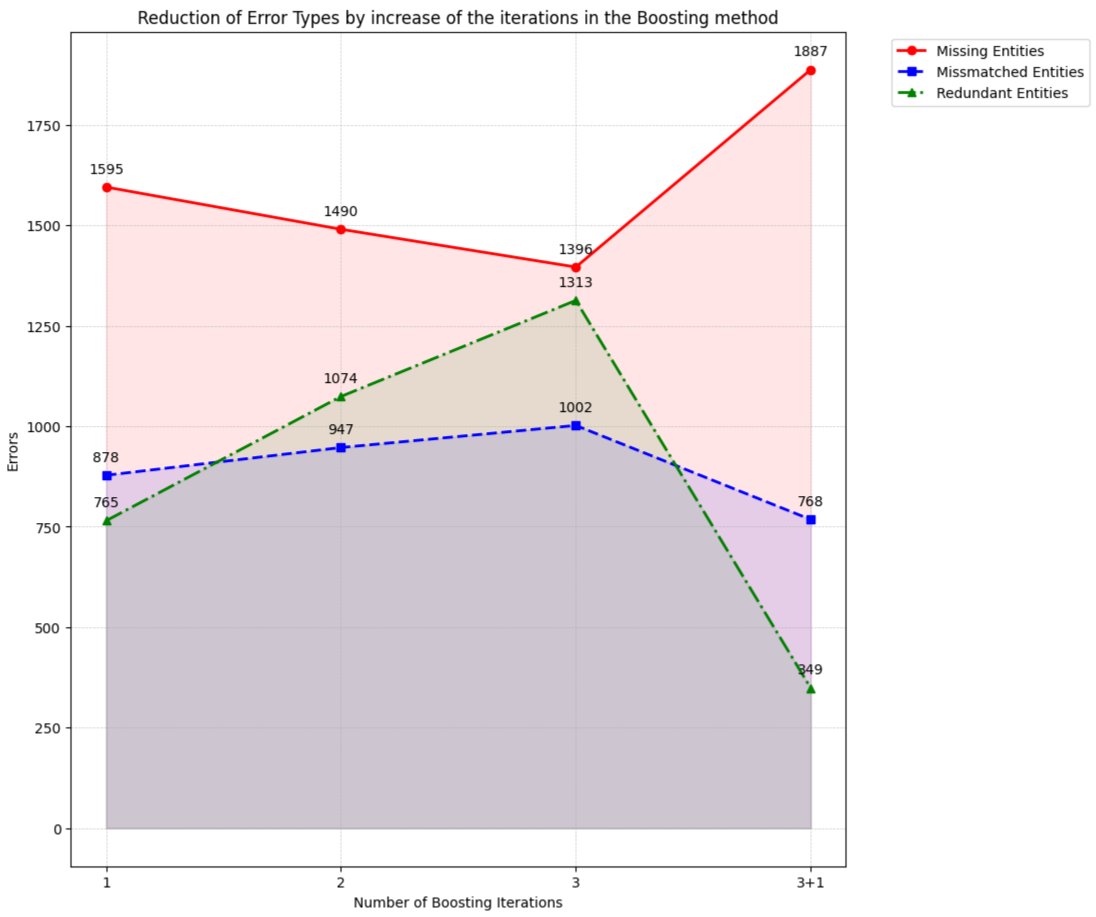
In addition to the previous figures, line charts for the individual methods were created to showcase the trend that occurs via increased iterations or voters. Looking at the bagging methodology reveals that the more voters participate in the process, the fewer FP occur, while FN remain mostly unchanged, as shown in Figure 13. This is due to the corrective nature of bagging, which tries explicitly to detect outliers and statistical anomalies. Only entities that occur in the majority of the voters are selected; this, therefore, reduces mismatched entities and redundant entities the most. Increasing the number of bagging voters from 3 to 7, meaning that an entity needed four votes to get in the final list, decreased the number of overall FP by 70. **Comparing this to the standard NER extraction, bagging led to a decrease of 145 FP (12%).** Similar results can be seen in the boosting methodology in Figure 14. Increasing the number of iterations led to a steady decrease in missing entities but also an increase in mismatched and redundant entities. This is so much so that the f1 score worsened as a result; this is because, in every iteration of boosting, the number of total entities that are pulled increases, which in turn leads to these results. Trying to add a corrective measure to aim to minimize the occurrence of FP, which can be seen as the last iteration in Figure 14, drastically returns the changes back to an even worse baseline.

Fig. 13: Line charts decrease of FP in the bagging approach by increasing iterations



Source: Own Results

Fig. 14: Line chart showing the change of errors in the boosting approach by increasing iterations



Source: Own Results

Increasing the number of boosting iterations from 1 to 3, meaning that there are more iterations over the extracted entities in an attempt to correct, decreased the number of overall missing entities by 199. **Comparing this to the standard NER extraction, boosting led to a decrease of 354 FN (20%).**

4.2 Feasibility Analysis

Taking these results into consideration, it is vital to assess how practical and viable they are when applied to real-world scenarios. In this section, the result data is evaluated and related to other key factors.

What becomes evident at first glance when looking at Figure 11 is that this is not a feasible approach to improve overall NER extraction performance. In the case of boosting, even when not considering other factors, it led to an overall decline in performance compared to standard NER extraction methods alone. The situation becomes even less favorable when accounting for the additional time and cost required for each iteration or vote. Every extra iteration effectively doubles the processing time and cost, with the final boosting iteration requiring four times the resources compared to a single standard run. Similarly, with the largest bagging configuration, the resource demand increased to seven times the standard amount. **Taking into account these considerations, the compromise of incurring sevenfold costs and processing duration for a mere 0,7% enhancement in overall performance is not feasible.**

There is, however, an argument that these methodologies may be relevant for specific use cases, especially in contexts where it is crucial to mitigate a specific category of hallucination. The bagging approach appears to hold potential, as it has demonstrated a reduction in FP (12%) without a corresponding rise in FN, contrasting with the outcomes observed in boosting techniques. The boosting methodology, however, demonstrated a much larger decrease (20%) in FN, with less than half the costs of the bagging technique, only needing three iterations.

4.3 Limitations

While the experiment yielded valuable insights, it is crucial to acknowledge the limitations that may have influenced the results. The major challenge faced throughout the process was the dataset itself. Despite limiting the data to the human-labeled portion, it was still riddled with inconsistencies. Two columns, misc and num, had to be dropped entirely, as they contained little to no discernible structure. Additionally, the remaining columns often lacked consistent uniformity. For example, names were sometimes preceded by titles or affixes such as "Dr." or "Sir," while in other instances, these were omitted.

Similarly, dates that were separated by a dash (e.g., "2014-2015") were sometimes split into two entities, like "2014" and "15," and at other times treated as a single entity. This kind of ambiguity permeated the entire dataset. These inconsistencies primarily led to mismatching errors, which formed a significant portion of the FP errors. This, in turn, decreased the overall performance of every approach on the precision score. This especially affected the bagging approach since its primary mechanism was the reduction of FP. **Therefore, despite bagging decreasing the number of FP errors by 12%, the full extent of this reduction remains unclear due to the pervasive inconsistencies in the data.** Nonetheless, it is also crucial to recognize that the primary error lies in the missing entities, as shown in Figure 12, which the inconsistencies present in the dataset did not influence. Additionally, the redundant entities were similarly not impacted by these discrepancies.

5. Conclusion

The following conclusion of this work reflects on the essential findings and the broader implications of the experiment while also considering avenues for future exploration and improvement.

This thesis set out to explore the potential of ensemble learning methods, specifically bagging and boosting, to mitigate hallucinations and improve the performance of LLMs in NER tasks. The experiment revealed that both methodologies resulted in performance scores nearly identical to standard NER extraction without providing substantial overall improvements. However, a closer inspection of the results demonstrated nuanced benefits. Boosting was more effective in reducing FN, lowering the number of missing entities by 20% (354 fewer FN), while bagging reduced FP more, decreasing mismatched and redundant entities by 12% (145 fewer FP). However, despite these improvements in specific error categories, the overall gain in performance was marginal, with only a 0,7% f1-score increase for bagging. This modest improvement came at the cost of a sevenfold increase in processing time and resources, making ensemble learning techniques impractical for general and real-world applications. Nevertheless, these methods may hold value in specialized use cases where it is important to target specific types of errors. That said, the full extent of these benefits remains somewhat uncertain due to the inconsistencies in the dataset, which complicates a more precise assessment of the true impact on performance.

While the experiment showed limited overall performance improvements when using bagging and boosting for NER tasks, the specific reduction in FN and FP highlights a potential of ensemble learning methods for targeted applications. This is noteworthy in domains where accuracy in particular types of entities is crucial, where reducing certain types of errors could have substantial implications. For instance, in contexts where missing critical information could lead to costly mistakes, such as an FN in a legal document omitting a key entity, boosting's ability to reduce such errors could prove practical. Similarly, bagging's effectiveness in reducing redundant or mismatched entities could play a role in fields where data accuracy and consistency are critical. Moreover, these findings raise further questions about how different error categories affect LLM usability in various domains and encourage deeper exploration into context-specific optimization strategies. Also, this research mainly investigated a general improvement in reducing hallucinations using NER as a control mechanism. The thesis looked at the default model of GPT-4o-mini since the emphasis was more on ensemble methods and a methodology for the overall improvement of hallucinations. Fine-tuning specific parameters and further prompt engineering could yield greater results for NER tasks.

This also raises important questions that call for further investigation. The experiment's findings suggest that ensemble methods could offer targeted solutions for reducing FP and FN in specialized contexts. However, the broader applicability of these techniques across different LLM tasks remains to be determined. Future research could explore how ensemble learning performs in other domains of language processing beyond NER, such as text summarization, machine translation, or sentiment analysis, where hallucinations and inaccuracies manifest differently. Additionally, further studies could investigate how variations in dataset quality, size, and diversity impact the effectiveness of these techniques, as the inconsistencies in the dataset used here limited a full assessment of the true potential of bagging and boosting. Finally, examining how ensemble techniques can be fine-tuned or combined with other emerging correction mechanisms to optimize performance for specific use cases would help advance this research and expand its real-world applicability. It also opens up the possibility of combining multiple techniques, each specialized in addressing specific errors. The experiment also revealed an interesting trade-off between the number of voters or iterations and the overall performance. This suggests that methods relying on multiple iterations can provide a form of confidence interval for system reliability. Hypothetically, in lower-stakes applications, where marginal inaccuracies are more acceptable, fewer iterations may still yield satisfactory results with reduced costs and processing time. However, in higher-stakes systems, such as those used in industries like medicine, where absolute precision and security are critical, more iterations would be necessary to ensure a higher confidence level and minimize the risk of errors. This would mirror existing practices in other fields, where redundant systems and rigorous testing are employed to guarantee safety and reliability. Therefore, the scalability of some techniques could offer flexibility, allowing them to be adapted to different risk levels and performance needs based on the specific context.

Bibliography

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., . . . Zoph, B. (2023). *GPT-4 Technical Report*. arXiv. Retrieved from http://arxiv.org/pdf/2303.08774

Ahmad, M. A., Yaramis, I., & Roy, T. D. (2023). *Creating Trustworthy LLMs: Dealing with Hallucinations in Healthcare AI*. arXiv. Retrieved from https://arxiv.org/abs/2311.01463

Allen, J. (1995). *Natural language understanding* (2nd ed.). Redwood City, California: The Benjamin/ Cummings Publ. Co., Inc.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv. Retrieved from http://arxiv.org/pdf/1409.0473

Bally, C., & Sechehaye, A. (Eds.) (1986). *Course in general linguistics* (5th ed.). Chicago: Open Court.

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (pp. 26–33).

Bender, M. E. (2021). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *FAccT '21: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610–623).

Bommasani, R., Soylu, D., Liao, I. T., Creel, A. K., & Liang, P. (2023). *Ecosystem Graphs: The Social Footprint of Foundation Models*. arXiv. Retrieved from https://arxiv.org/pdf/2303.15772

Brill, E., & Raymond, J. M. (1997). An Overview of Empirical Natural Language Processing. *AI Magazine*, *18*(4), 13–24.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. In *NIPS '20: Proceedings of the 34th International Conference on Neural Information Processing Systems* (pp. 1877–1901).

Camacho-Collados, J., & Pilehvar, M. T. (2020). *Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning* (1st ed.). San Rafael, California: Morgan & Claypool Publishers.

Carr, D. F. (2023, February 3). ChatGPT Tops 25 Million Daily Visits. Retrieved from https://www.similarweb.com/blog/insights/ai-news/chatgpt-25-million/

Chang, H.-S., & McCallum, A. (2022). Softmax Bottleneck Makes Language Models Unable to Represent Multi-mode Word Distributions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 8048–8073).

Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2023). *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. arXiv. Retrieved from http://arxiv.org/pdf/2310.14735

Chiarello, F., Giordano, V., Spada, I., Barandoni, S., & Fantoni, G. (2024). Future applications of generative large language models: A data-driven case study on ChatGPT. *Technovation*, *133*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Deep Learning and Representation Learning Workshop*.

Chomsky, N. (2020). *Syntactic Structures* (12th ed.). Berlin, Boston: De Gruyter Mouton.

Dahl, M., Magesh, V., Suzgun, M., & Ho, D. E. (2024). Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models. *Journal of Legal Analysis*, *16*(1), 64–93.

Dasari, A. K., Biswas, S. K., Thounaojam, D. M., Devi, D., & Purkayastha, B. (2023). Ensemble Learning Techniques and Their Applications: An Overview. In *International Conference on Communications and Cyber Physical Engineering 2018* (pp. 897–912).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019* (pp. 4171–4186).

Eloundou, T., Manning, S., Mishkin, P., & Rock, D. (2023). *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*. arXiv. Retrieved from http://arxiv.org/pdf/2303.10130

Fathi, E., & Maleki Shoja, B. (2018). Chapter 9 - Deep Neural Networks for Natural Language Processing. *Handbook of Statistics*, *38*, 229–316.

Fernández, S., Graves, A., & Schmidhuber, J. (2007). An Application of Recurrent Neural Networks to Discriminative Keyword Spotting. In *Artificial Neural Networks - ICANN 2007, 17th International Conference* (pp. 220–229).

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Gairola, A. (2024, June 12). Google Engineer Says OpenAI Set Back AI Research by 5-10 Years Due to LLM Hype. *Benzinga*. Retrieved from https://www.benzinga.com/news/24/06/39284426/google-engineer-says-sam-altman-led-openai-set-back-ai-research-progress-by-5-10-years-llms-have-suc

Gekhman, Z., Yona, G., Aharoni, R., Eyal, M., Feder, A., Reichart, R., & Herzig, J. (2024). *Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?* arXiv. Retrieved from http://arxiv.org/pdf/2405.05904

Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., . . . Vinyals, O. (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. arXiv. Retrieved from http://arxiv.org/pdf/2403.05530

Ghose, S. (2024, May 2). Why Hallucinations Matter: Misinformation, Brand Safety and Cybersecurity in the Age of Generative AI. Berkeley. Retrieved from https://scet.berkeley.edu/why-hallucinations-matter-misinformation-brand-safety-and-cybersecurity-in-the-age-ofgenerative-ai/

Goodfellow, I., Courville, A., & Bengio, Y. (2016). *Deep learning*. Cambridge, Massachusetts: The MIT Press.

Grishman, R. (1986). *Computational linguistics: An introduction*. Cambridge: Cambridge Univ. Press.

Grosz, J. B., & Candace, L. S. (1986). ATTENTION, INTENTIONS, AND THE STRUCTURE OF DISCOURSE. *Computational Linguistics*, *12*(3), 175–204.

Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C. C. T., Giorno, A. D., Gopi, S., . . . Li, Y. (2023). *Textbooks Are All You Need*. arXiv. Retrieved from http://arxiv.org/pdf/2306.11644

Holtzman, A., Buys, J., Du Li, Forbes, M., & Choi, Y. (2020). The Curious Case of Neural Text Degeneration. *ICLR 2020 Conference*.

Huang, L., Weijiang, Y., Weitao, M., Weihong, Z., Zhangyin, F., Haotian, W., . . . Ting, L. (2023). *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. arXiv. Retrieved from https://arxiv.org/pdf/2311.05232

Hudry, J.-L. (2015). Aristotle on Language and Universal Proof. *The Road to Universal Logic*, 267–281.

Huet, E. (2024, May 17). AI Fake Bylines on News Site Raise Questions of Credibility for Journalists. *Bloomberg*. Retrieved from https://www.bloomberg.com/news/newsletters/2024-05-17/ai-fake-bylines-on-news-site-raise-questions-of-credibility-for-journalists

Irugalbandara, C., Mahendra, A., Daynauth, R., Arachchige, T. K., Dantanarayana, J., Flautner, K., . . . Mars, J. (2024). *Scaling Down to Scale Up: A Cost-Benefit Analysis of Replacing OpenAI's LLM with Open Source SLMs in Production*. arXiv. Retrieved from http://arxiv.org/pdf/2312.14972v3

Islam, R., & Moushi, O. M. (2024). *GPT-4o: The Cutting-Edge Advancement in Multimodal LLM*. TechRxiv. Retrieved from https://www.techrxiv.org/users/771522/articles/1121145-gpt-4o-the-cutting-edge-advancement-in-multimodal-llm

Islam, S., Elmekki, H., Elsebai, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. (2024). A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications*, *241*.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., . . . Fung, P. (2022). Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, *55*(12), 1–38.

Johri, P., Khatri, K. S., Al-Taani, T. A., Sabharwal, M., & Kumar, A. (2021). Natural Language Processing: History, Evolution, Application, and Future Work. In *Proceedings of 3rd International Conference on Computing Informatics and Networks* (pp. 365–375).

Jordan, M. I. (1997). Serial Order: A Parallel Distributed Processing Approach. *Advances in Psychology*, *121*, 471–495.

Jurafsky, D., & Martin, J. H. (2024). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Stanford. Retrieved from https://web.stanford.edu/~jurafsky/slp3/

Kaddari, Z., Mellah, Y., Berrich, J., Blkasmi, G. M., & Bouchentouf, T. (2021). Artificial intelligence and industrial applications: Artificial intelligence techniques for cyber-physical, digital twin systems and engineering applications. *Lecture Notes in Networks and Systems (LNNS)*, *144*, 236–246.

Kang, H., & Liu, X.-Y. (2023). *Deficiency of Large Language Models in Finance: An Empirical Examination of Hallucination*. arXiv. Retrieved from http://arxiv.org/pdf/2311.15548

Kang, H., Ni, J., & Yao, H. (2024). *Ever: Mitigating Hallucination in Large Language Models through Real-Time Verification and Rectification*. arXiv. Retrieved from https://arxiv.org/pdf/2311.09114

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., . . . Amodei, D. (2020). *Scaling Laws for Neural Language Models*. arXiv. Retrieved from http://arxiv.org/pdf/2001.08361

Komatsuzaki, A. (2020). *Current Limitations of Language Models: What You Need is Retrieval*. arXiv. Retrieved from http://arxiv.org/pdf/2009.06857

Lee, N., Ping, W., Xu, P., Patwary, M., Fung, P., Shoeybi, M., & Catanzaro, B. (2023). *Factuality Enhanced Language Models for Open-Ended Text Generation*. arXiv. Retrieved from https://arxiv.org/pdf/2206.04624

Leibniz, G. W. (1989). Dissertation on the Art of Combinations. *Philosophical Papers and Letters*, *2*, 73–84.

Lester, B., Al-Rfou, R., & Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 3045–3059).

Li, J., Chen, J., Ren, R., Cheng, X., Zhao, W. X., Nie, J.-Y., & Wen, J.-R. (2024). The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 10879–10899).

Li, Z., Zhang, S., Zhao, H., Yang, Y., & Yang, D. (2023). *BatGPT: A Bidirectional Autoregessive Talker from Generative Pre-trained Transformer*. arXiv. Retrieved from https://arxiv.org/pdf/2307.00360

Lipton, C. Z., Berkowitz, J., & Elkan, C. (2015). *A Critical Review of Recurrent Neural Networks for Sequence Learning*. arXiv. Retrieved from https://arxiv.org/pdf/1506.00019

Luo, L. (2023). Influence of Interface Design Driven by Natural Language Processing on User Participation. *Frontiers in Business Economics and Management*, *11*(3), 63–66.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., . . . Clark, P. (2023). *Self-Refine: Iterative Refinement with Self-Feedback*. arXiv. Retrieved from http://arxiv.org/pdf/2303.17651

Maleki, N., Padmanabhan, B., & Dutta, K. (2024). AI Hallucinations: A Misnomer Worth Clarifying. In *2024 IEEE Conference on Artificial Intelligence (CAI)* (pp. 133–138).

Maslej, N., Fattorini, L., Perrault, R., Parli, V., Reuel, A., Brynjolfsson, E., . . . Clark, J. (2024). *Artificial Intelligence Index Report 2024*. arXiv. Retrieved from http://arxiv.org/pdf/2405.19522

Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On Faithfulness and Factuality in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 1906–1919).

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

Metz, C. (2023, November 6). Chatbots May 'Hallucinate' More Often Than Many Realize. *The New York Times*. Retrieved from https://www.nytimes.com/2023/11/06/technology/chatbots-hallucination-rates.html

Miao, M., Meng, F., Liu, Y., Zhou, X.-H., & Zhou, J. (2021). Prevent the Language Model from being Overconfident in Neural Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics* (pp. 3456–3468).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)* (pp. 1–12).

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., . . . Mian, A. (2024). *A Comprehensive Overview of Large Language Models*. arXiv. Retrieved from http://arxiv.org/pdf/2307.06435

OpenAI (2024). Models: Models overview. Retrieved from https://platform.openai.com/docs/models/gpt-4o-mini

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., . . . Lowe, R. (2011). Training language models to follow instructions with human feedback. In *NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems* (pp. 27730–27744).

Pakhale, K. (2023). *Comprehensive Overview of Named Entity Recognition: Models, Domain-Specific Applications and Challenges*. arXiv. Retrieved from http://arxiv.org/pdf/2309.14084

Patel, D., & Wong, G. (2023, July 10). GPT-4 Architecture, Infrastructure, Training Dataset, Costs, Vision, MoE. *SemiAnalysis*. Retrieved from https://www.semianalysis.com/p/gpt-4-architecture-infrastructure

Patil, R., & Gudivada, V. (2024). A Review of Current Trends, Techniques, and Challenges in Large Language Models (LLMs). *Applied Sciences*, *14*(5).

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).

Pham, D. K., & Vo, B. Q. (2024). *Towards Reliable Medical Question Answering: Techniques and Challenges in Mitigating Hallucinations in Language Models*. arXiv. Retrieved from http://arxiv.org/pdf/2408.13808v1

Radford, A., & Narasimhan, K. (2018). *Improving Language Understanding by Generative Pre-Training*. OpenAI. Retrieved from https://cdn.openai.com/research-covers/language-unsuper-vised/language_understanding_paper.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. OpenAI. Retrieved from https://cdn.openai.com/better-lan-guage-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., . . . Liu, P. J. (2020). Explor-ing the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *The Journal of Ma-chine Learning Research*, *21*(1), 5485–5551.

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26 th International Conference on Machine Learning* (pp. 873–880).

Ramanathan, T. (2024, June 17). natural language processing. Encyclopedia Britannica. Retrieved from https://www.britannica.com/technology/natural-language-processing-computer-science

Ray, P. P. (2023). ChatGPT: A comprehensive review on background, applications, key chal-lenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Sys-tems*, *3*, 121–154.

Rhys Leahy, R. F. S., J. Restrepo, N., Lupu, Y., & F. Johnson, N. (2021). Machine Learning Lan-guage Models: Achilles Heel for Social Media Platforms and a Possible Solution. *Advances in Artificial Intelligence and Machine Learning*, *1*(3), 191–202.

Rumelhart, D., Geoffrey, E. H., & Ronald, J. W. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, *1*, 318–362.

Sutskever, I., Vinyals, O., & Le V, Q. (2014). *Sequence to Sequence Learning with Neural Net-works*. arXiv. Retrieved from http://arxiv.org/pdf/1409.3215

Taplin, S. (2024, August 15). AI Hallucinations: How Can Businesses Mitigate Their Impact? *Forbes*. Retrieved from https://www.forbes.com/councils/forbestechcouncil/2024/08/15/ai-hallucinations-how-can-businesses-mitigate-their-impact/

Ting, K. M. (2011). Confusion Matrix. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (p. 209). Boston, MA: Springer US.

Tonmoy, S. M. T. I., Zaman, S. M. M., Jain, V., Rani, A., Rawte, V., Chadha, A., & Das, A. (2024). *A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models*. arXiv. Retrieved from https://arxiv.org/pdf/2401.01313

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., . . . Lample, G. (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv. Retrieved from http://arxiv.org/pdf/2302.13971

TURING, A. M. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, *LIX*(236), 433–460.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. In *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 6000–6010).

Wang, C., & Sennrich, R. (2020). On Exposure Bias, Hallucination and Domain Shift in Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 3544–3552).

Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., . . . Wang, G. (2023). *GPT-NER: Named Entity Recognition via Large Language Models*. arXiv. Retrieved from https://arxiv.org/pdf/2304.10428

Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., . . . Gabriel, I. (2021). *Ethical and social risks of harm from Language Models*. arXiv. Retrieved from http://arxiv.org/pdf/2112.04359

Weizenbaum, J. (1966). ELIZA-a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Wilks, Y. (1975). A preferential, pattern-seeking, Semantics for natural language inference. *Artificial Intelligence*, *6*(1), 53–74.

Wong, M. (2023, February 16). Bing and Google's chatbots are a disaster. *The Atlantic*. Retrieved from https://www.theatlantic.com/technology/archive/2023/02/google-microsoft-search-engine-chatbots-unreliability/673081/

Wonhyeong, S. (2022). Normalization and pre-tokenization. Hugging Face. Retrieved from https://huggingface.co/learn/nlp-course/chapter6/4

Yadlin, A., & Marciano, A. (2024). Hallucinating a political future: Global press coverage of human and post-human abilities in ChatGPT applications. *Media, Culture & Society*, *0*(0).

Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., . . . Sun, M. (2019). DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 764–777).

Zhang, L., Liu, X., Li, Z., Pan, X., Dong, P., Fan, R., . . . Chu, X. (2023). *Dissecting the Runtime Performance of the Training, Fine-tuning, and Inference of Large Language Models*. arXiv. Retrieved from https://arxiv.org/pdf/2311.03687

Zhou, Y., Cui, C., Yoon, J., Zhang, L., Deng, Z., Finn, C., . . . Yao, H. (2024). Analyzing and Mitigating Object Hallucination in Large Vision-Language Models. *ICLR 2024*.

Zhou, Z.-H. (2012). *Ensemble methods: Foundations and algorithms* (1st ed.). New York: Chapman and Hall/CRC.

Appendices

Appendix directory

Appendix A: Experiment Results

ner_score_table

Method	True Positives	False Positives	False Negatives	Precision	Recall	F1
Standard	5577	1178	1750	82,6%	76,1%	79,2%
Bagging	5638	1117	1887	83,5%	74,9%	79%
Boosting	5618	1033	1791	84,4%	75,8%	79,9%

ner_error_table

Method	Missing Entitites	Missmatched Entitites	Redundant Entities
Standard	5577	1178	1750
Bagging	5638	1117	1887
Boosting	5618	1033	1791

boosting_error_table

Boosting Iterations	Missing Entitites	Missmatched Entitites	Redundant Entities
1	1595	878	765
2	1490	947	1074
3	1396	1002	1313
3+1	1887	768	349

boosting_score_table

Boosting Iterations	TP	FP	FN	Precision	Recall	F1
1	5787	1643	1595	77,9%	78,4%	78,1%
2	5881	2021	1490	74,4%	79,8%	77%
3	5937	2315	1396	71,9%	81%	76,2%
3+1	5638	1117	1887	83,5%	74,9%	79%

bagging_error_table

Bagging Votes	Missing Entitites	Missmatched Entitites	Redundant Entities
3	1795	712	390
5	1791	702	369
7	1791	672	361

bagging_score_table

Bagging Votes	TP	FP	FN	Precision	Recall	F1
3	5584	1102	1795	83,5%	75,7%	79,4%
5	5613	1071	1791	84%	75,8%	79,7%
7	5618	1033	1791	84,5%	75,8%	79,9%

Appendix B: Prompts

standard_ner_extract

Prompt 1:

“You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text and return their positions based on the tokenized version of the text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015)

You will receive two inputs:

1. Organic text: The natural text for the sentence.

2. Tokenized text: A list of tokens representing the original text.

You should extract named entities from the organic text and return their start and end token positions based on the tokenized text. The positions should be returned as lists of start and end indices.

Example:

Organic Text: 'John Doe went to New York on September 5, 2024.'

Tokenized Text: [['John', 'Doe'], ['went'], ['to'], ['New', 'York'], ['on'], ['September', '5', ',', '2024']]

Entities: ['{PER': 'John Doe'}, {'LOC': 'New York'}, {'TIME': 'September 5, 2024'}]

Token Positions: [[0, 1], [3, 4], [5, 7]]”

Prompt 2:

“You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015)

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example:

[{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."

Validated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

Prompt 3:

“You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015) (TIME entities seperated by "–" and "-"are to be treated as seperate entities).

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example:

[{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."

Validated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

boosting_ner_extract

Prompt 1:

Extract:

You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015)

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example:

[{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."

Validated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

Verification False Negatives:

You are an NER entity validation and correction system. You will be given the original text and the extracted entities. Your task is to find any missing entities that were not identified in the first extraction and to make sure every possible entity in the text is correctly classified as PER (person), ORG (organization), LOC (location), or TIME (date or period). Focus specifically on entities that might have been missed in the first pass. Do not change or remove the entities that are already correct unless absolutely necessary. If you find any new entities in the text, make sure to add them to the existing list in the correct format. The output should include both the previously identified entities and the new entities, if any. The output should always return the validated entities in the exact same structured list format as showcased in the example.

Example:

Validated Entities: [{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]

Verification False Positives:

You are an NER entity validation and correction system. You will be given the original text and a list of extracted entities. Your task is to identify any incorrect or irrelevant entities that have been mistakenly included in the second extraction. Focus specifically on detecting entities that do not fit the categories of PER (person), ORG (organization), LOC (location), or TIME (date or period) or were incorrectly classified. If any extracted entities are unnecessary or improperly labeled, remove or correct them. Ensure that all remaining entities are correctly classified and necessary. Do not add new entities in this step—focus only on removing or fixing those that are wrongly included. The output should always return the validated entities in the exact same structured list format as showcased in the example.

Example:

Validated Entities: [{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]

Prompt 2:

Extract:

You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015)

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example:

[{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."

Validated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

Verification False Negatives:

You are an NER entity validation and correction system. You will be given the original text and the extracted entities. Your task is to find any missing entities that were not identified in the first extraction and to make sure every possible entity in the text is correctly classified as PER (person), ORG (organization), LOC (location), or TIME (date or period). Focus specifically on entities that might have been missed in the first pass. Do not change or remove the entities that are already correct unless absolutely necessary. If you find any new entities in the text, make sure to add them to the existing list in the correct format. The output should include both the previously identified entities and the new entities, if any. The output should always return the validated entities in the exact same structured list format as showcased in the example.

Example:

Validated Entities: [{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]

Verification False Positives:

You are an NER entity validation and correction system. You will be given the original text and a list of extracted entities. Your task is to identify any incorrect or irrelevant entities that have been mistakenly included in the second extraction. Focus specifically on detecting entities that do not fit the categories of PER (person), ORG (organization), LOC (location), or TIME (date or period) or were incorrectly classified. If any extracted entities are unnecessary or improperly labeled, remove or correct them. Ensure that all remaining entities are correctly classified and necessary. Do not add new entities in this step—focus only on removing or fixing those that are wrongly included. The output should always return the validated entities in the exact same structured list format as showcased in the example.

Example

:Validated Entities: [{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]

Prompt 3:

Verification False Negatives:

You are an NER entity validation and correction system. You will be given the following:

1. The original text.

2. A list of extracted entities.

Your task is to:

1. Find and add any missing entities that belong to one of the categories: PER (person): Names of individuals, ORG (organization): Names of companies, institutions, or groups, LOC (location): Names of cities, countries, or places, TIME (time period): Specific dates, years, or times (TIME entities seperated by "–" and "-"are to be treated as seperate entities)

2. Do not change or remove any correct entities unless they are obviously incorrect or misclassified.

3. Only add new entities if they clearly belong to one of the categories above.

Make sure that the output is always returned in the following structured format:

Validated Entities: [{"PER": "François Asselineau", {"LOC": "Eurozone"}, {"ORG": "BBC1"}, {"TIME": "twelve years"}]

Prompt 4:

Extract:

You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015) (TIME entities seperated by "–" and "-"are to be treated as seperate entities).

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example:

[{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."\nValidated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

Verification False Negatives:

You are an NER entity validation and correction system. You will be given the following:

1. The original text.

2. A list of extracted entities.

Your task is to:

1. Find and add any missing entities that belong to one of the categories: PER (person): Names of individuals, ORG (organization): Names of companies, institutions, or groups, LOC (location): Names of cities, countries, or places, TIME (time period): Specific dates, years, or times (TIME entities seperated by "–" and "-"are to be treated as seperate entities).

2. Do not change or remove any correct entities unless they are obviously incorrect or misclassified.\n3. Only add new entities if they clearly belong to one of the categories above.

Make sure that the output is always returned in the following structured format:\nValidated Entities: [{"PER": "François Asselineau"}, {"LOC": "Eurozone"}, {"ORG": "BBC1"}, {"TIME": "twelve years"}]

Verification False Positives:

You are an NER entity validation and correction system. You will be given the following:

1. The original text.

2. A list of extracted entities.

Your task is to:

1. Identify any incorrect or irrelevant entities that have been mistakenly included in the extraction. Focus specifically on detecting entities that do not fit the categories of PER (person): Names of individuals, ORG (organization): Names of companies, institutions, or groups, LOC (location): Names of cities, countries, or places, TIME (time period): Specific dates, years, or times (TIME entities seperated by "–" and "-"are to be treated as seperate entities), or were incorrectly classified.

2. If any extracted entities are unnecessary or improperly labeled, remove or correct them. Ensure that all remaining entities are correctly classified and necessary.

3. Do not add new entities in this step, focus only on removing or fixing those that are wrongly included.

Make sure that the output is always returned in the following structured format:

Validated Entities: [{"PER": "Stan Penridge"}, {"LOC": "Luxembourg"}, {"ORG": "Jin dynasty"}, {"TIME": "1991"}]

bagging_ner_extract

Prompt 1:

You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015)

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example: [{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."

Validated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

Prompt 2:

You are a named entity recognition (NER) system. Your job is to extract all named entities from the provided text. The entities that you should identify are:

- PER for people (e.g., Bergqvist, Kee Marcello)

- ORG for organizations (e.g., AirAsia Zest, Civil Aviation Authority of the Philippines)

- LOC for geographic locations (e.g., Ninoy Aquino International Airport, Manila, Indian Ocean)

- TIME for dates and times (e.g., 2013, August 16, 2013, late 2015) (TIME entities seperated by "–" and "-"are to be treated as seperate entities).

Provide the result in a structured list format in the same order as you encountered the entities in the sentence, the single entities should be stored as dictionaries with the key being the type such as PER and the value the indentified entity.

Example:

[{"ORG": "AirAsia"}, {"LOC": "Manila"}, {"TIME": "2013"}]. Always adhere to this format, do not use any other format, even if there is only one entity type.

Example:

Text: "John Doe went to New York on September 5, 2024."

Validated Entities: [PER: John Doe, LOC: New York, TIME: September 5, 2024]

Appendix C: Python Code

The full Python code for the experiment can be found on Github:

https://github.com/sanax-997/Performance-improvement-of-LLMs-for-Named-Entity-Recognition-using-Ensemble-Learning-techniques.git

Affidavit

EIDESSTATTLICHE ERKLÄRUNG

INTERNATIONALE HOCHSCHULE

Hiermit versichere ich an Eides statt, dass ich die Abschlussarbeit selbständig und ohne Inanspruchnahme fremder Hilfe angefertigt habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen. Ich erkläre mich damit einverstanden, dass die Arbeit mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft wird.

Neufeld, 22.10.2024	[illegible]
Ort, Datum	Unterschrift