Daniel Arias Cámara

From Traditional to Large Language Models: A Novel NLP-Based Model for Sentiment Analysis in Social Media

University Master's Degree in Computer Security Engineering and Artificial Intelligence

Final Master's Thesis

Directed by

Dr. Jordi Pascual Fontanilles

UNIVERSITAT ROVIRA i VIRGILI

Tarragona, 2025

Contents

List of Figures and Tables

Figures

Tables

Abstract

This thesis addresses the need for Catalan sentiment analysis tools by creating a new 23,000-sample balanced corpus and developing a specialized classification model, the CSXSC. This fine-tuned, 125M-parameter, encoder-only RoBERTa model was benchmarked against two 7-billion-parameter decoder-only models trained efficiently using QLoRA. The results demonstrate the superiority of the specialized model, which achieved a final test set accuracy of 83.69% while being over 24 times more computationally efficient. This study concludes that for this discriminative task, a smaller, architecturally appropriate model provides a more accurate and practical solution than larger, general-purpose LLMs.

"Language is a collective human creation, reflecting human nature, how we conceptualize reality, how we relate to one another."

- Stephen Pinker

1 Introduction

1.1 The origins of language

Homo sapiens, meaning wise man in Latin and commonly known as human being, is the latest species of the genus *Homo*. This name reflects the defining characteristic of our species: an unparalleled capacity for reasoning, problem-solving, and complex communication. Unlike any known animal communication system, the human language is a powerful and flexible tool that enables us to convey an infinite range of ideas. More than a means of exchanging simple information, language shapes our thoughts, strengthens our memories, and allows us to build civilizations. It is through language that we transmit knowledge across generations, preserve cultural traditions, and explore the mysteries of science and philosophy [1].

The origins of language remain one of the greatest mysteries in evolutionary science. Estimates suggest that spoken language emerged around 100,000 years ago, while writing appeared much later, approximately 5,000 years ago [2]. However, some studies challenge the idea of sudden emergence, instead proposing that language evolved gradually over millions of years through a continuous adaptation and refinement process [3]. Regardless of the precise timeline, it is clear that language provided *Homo sapiens* with an unprecedented evolutionary advantage, allowing our species to organize, innovate and ultimately dominate the planet.

Throughout history, language has continually evolved, adapting to the changing needs and complexities of human societies. According to *Ethnologue*, a comprehensive reference work cataloging all known living languages in the world, there are currently 7,159 officially recognized languages [4], although this number is decreasing as linguistic communities decline [5]. This rich tapestry of languages reflects the numerous ways humans have found to express thoughts, emotions, and knowledge, underscoring the adaptability and creativity inherent in our species.

1.2 Language Beyond Human-to-Human Communication

The idea of using language to communicate with machines goes back to Alan Turing, a pioneer in artificial intelligence. In 1950, he proposed a test, known as *Turing test*, to see if a machine could think like a human [6]. The test works by having a person chat with both a machine and another human without knowing which is which. If the machine can respond in a way that makes it impossible to tell it apart from a human being, it is considered "intelligent." However, many researchers have questioned whether this test really proves intelligence. Some argue that it only shows how well a machine can imitate human conversation, not whether it truly understands or thinks [7]. Despite these doubts, the Turing test remains an important idea in artificial intelligence, as it highlights how crucial language is in measuring intelligence.

Large Language Models (LLMs), like ChatGPT and Gemini, are advancing rapidly, making the idea of machines passing the Turing test more realistic. These AI systems learn from large amounts of text and can generate responses that sound almost human [8]. Even though they don't truly understand or think like we do, their ability to imitate human conversation is improving so much that even experts may struggle to tell the difference between AI and a real person [9]. If LLMs consistently pass the Turing test, it will challenge how we define intelligence and make us reconsider our relationship with AI in the future.

1.3 Evolution of Large Language Models

To track the evolution of intellectual capabilities in LLMs, Rein et al. [10] introduced the *Graduate-level Google-Proof Q&A* (GPQA) questionnaire in 2023. Designed to test advanced reasoning, GPQA goes beyond pattern matching and shallow retrieval by requiring deep understanding. It features challenging science questions at the level of Ph.D. qualifying exams across disciplines like physics, chemistry, biology, and computer science. Its "Google-Proof" design ensures that answers cannot be easily found online, making it a reliable tool to evaluate true comprehension and reasoning in LLMs.

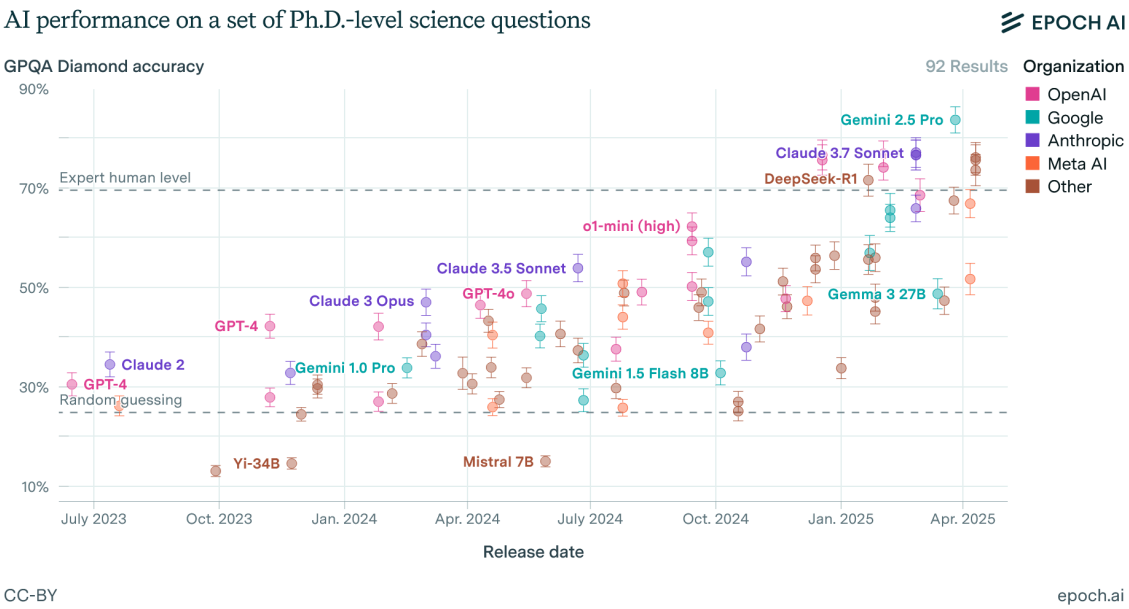

Figure 1: Performance of AI systems on GPQA Diamond in recent years

Extracted from EPOCH AI[1], Figure 1 illustrates the performance of various state-of-the-art LLMs in the GPQA Diamond subset and a clear upward trend can be observed in their reasoning capabilities. Models released in early 2023, such as GPT-4 and Claude 2,

[1] https://epoch.ai/data/ai-benchmarking-dashboard

showed moderate performance, typically between 30% and 40%. However, more recent models, including Claude 3.5 Sonnet, DeepSeek-R1, and Gemini 2.5 Pro, have surpassed the *Expert human level*, reaching accuracies above 70%. This suggests that AI systems are not only improving in language fluency, but also making significant strides in scientific reasoning and problem-solving capabilities.

A key factor behind the remarkable performance of modern LLMs is the enormous computational effort required to train them. This is commonly quantified in *Floating-Point Operations* (FLOPs), which measure the total number of arithmetic operations performed during training [11]. As demonstrated by Kaplan et al. [12], models with higher FLOP counts generally require more data, computational resources, and training time, but also tend to achieve better generalization and exhibit stronger reasoning abilities, following predictable scaling laws.

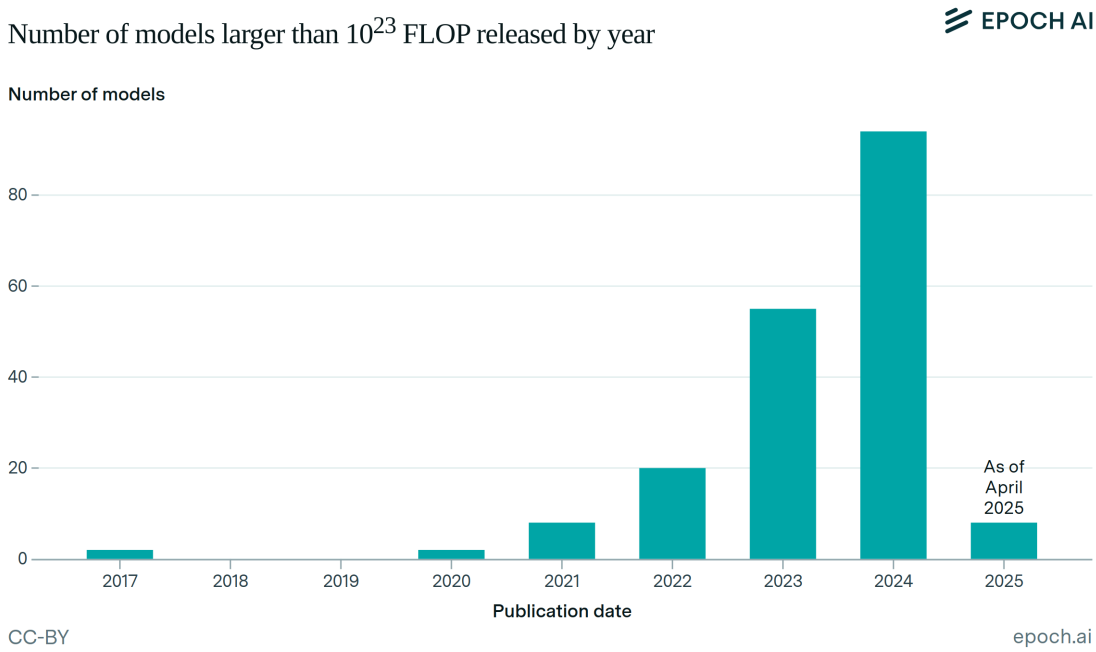

Figure 2: Number of models larger than 10^{23} FLOP released by year

Figure 2 illustrates the growing number of models exceeding 10^{23} FLOPs released each year. Until 2021, only a handful of such large models existed. However, from 2022 onward, there has been a steep increase. In 2023, over 50 models surpassed the 10^{23} FLOP threshold, and this number surged to nearly 90 in 2024. This trend reflects the rapid acceleration in LLM development, driven by advances in hardware and increased investment in AI research and deployment.

1.4 Natural Language Processing

Although LLMs seem to understand and produce human-like language, at their core, they operate by tokenizing text, breaking it down into smaller, more manageable units known as tokens, which may include words, subwords, or characters. What allows these models to interact meaningfully with human language are the principles and methodologies rooted in the field of *Natural Language Processing* (NLP). Natural Language Processing is a branch of artificial intelligence focused on analyzing and understanding linguistic data, most commonly textual documents, using computational techniques. Its primary goal is to construct structured representations of unstructured natural language leveraging insights from linguistics and computer science [13].

The development of NLP has evolved through several distinct phases. The first approaches, known as *symbolic methods* or rule-based NLP, relied on explicitly defined linguistic rules and formal grammars. These systems were designed to simulate human understanding by manipulating language through logic and symbolic representations. A notable example is the *Conceptual Dependency Theory* proposed by Roger Schank in the 1970s, which attempted to model meaning based on abstract representations of actions and entities [14]. However, critics such as philosopher John Searle argued that such rule-based manipulation lacked true understanding, famously illustrated through the *Chinese Room* thought experiment [15], which questioned whether symbolic processing alone could ever lead to genuine comprehension.

The *Chinese Room* experiment, proposed by John Searle in 1980, imagines a person who does not understand Chinese locked in a room. This person is given a set of Chinese symbols along with a rulebook (written in their native language) that allows them to match input symbols with appropriate output symbols. From the outside, it appears as if the person understands Chinese, because the responses are appropriate and coherent. However, the person is merely manipulating symbols based on rules, without any understanding of their meaning. Searle used this analogy to argue that computers similarly may appear to understand language when following programmed rules, but they do not possess true comprehension or consciousness.

In the 1990s, NLP entered a new phase with the rise of *statistical methods*. Due to the increasing availability of digital advances in computing power, researchers began using probabilistic techniques to uncover patterns in language data. Models such as *Hidden Markov Models* (HMMs) [16] and *Conditional Random Fields* (CRFs) [17] became foundational for tasks such as part-of-speech tagging and named entity recognition. The seminal work of Manning and Schütze [18] formalized many of these ideas and established statistical NLP as a robust, data-driven alternative to rule-based systems.

The most significant breakthrough came in 2017 with the introduction of the *Transformer* architecture by researchers at Google [19]. Unlike previous approaches, Transformers utilize a *self-attention mechanism* [20] that allows models to evaluate relationships between all words in a sentence simultaneously, regardless of their position. This innovation greatly improved the ability of models to understand long-range dependen-

cies in text and improved training efficiency. Transformers soon became the backbone of modern NLP, leading to the development of powerful pre-trained models such as BERT [21] or GPT [22].

1.5 Sentiment Analysis and Social Media

Natural Language Processing covers a wide range of real-world applications, including chatbots and virtual assistants, search engines, machine translation, text classification, information extraction, text summarization, plagiarism detection, market and financial analysis, and recommendation systems [23]. Among these applications, sentiment analysis plays a prominent role. Identifying emotions, attitudes, or opinions expressed in textual data, such as product reviews or social media posts, enables systems to interpret subjective information effectively.

The importance of sentiment analysis has grown considerably due to its relevance across various domains. It serves as a fundamental tool in business intelligence, global financial markets, smart home technologies, and mental health monitoring. For example, it facilitates the detection of hate speech, the recognition of emotional signals in suicide notes, and the evaluation of stress levels through language patterns [24]. These capabilities highlight the social and commercial significance of sentiment analysis as a subfield within NLP.

The emergence of deep learning, particularly the development of LLMs, has transformed the landscape of sentiment analysis. Traditional techniques have increasingly been replaced by models capable of learning directly from large-scale textual corpora. LLMs such as GPT-3 [25], LaMDA [26], and Bloom [27] exemplify this shift, utilizing advanced transformer-based architectures with billions of parameters to model language in a highly contextualized and nuanced manner [28]. These models exhibit strong abilities to identify sentiment across diverse expressions, including subtle, sarcastic, or ambiguous language, and can generalize across domains with minimal task-specific adjustments. Consequently, LLMs have demonstrated superior performance in a range of sentiment analysis tasks, from polarity detection to emotion recognition.

Moreover, the integration of LLMs into sentiment analysis systems has advanced the field toward a more human-like understanding of language. Their ability to generalize from heterogeneous and multilingual datasets makes them particularly suitable for analyzing informal or code-mixed language, which is common in real-world communications. This is especially relevant in the context of social media, where user-generated content often includes slang, abbreviations, and highly contextual language [29]. As LLMs continue to evolve, they are expected to drive new breakthroughs in sentiment analysis research, expanding their impact across academic, industrial, and society applications.

2 Natural Language Processing

This chapter introduces the fundamental concepts of *Natural Language Processing* (NLP). The purpose of this chapter is to provide the reader with a comprehensive understanding of how natural language is represented and interpreted by machines, the two main approaches to understand and generate text, and the real-world applications, challenges, and future research directions in the field of NLP.

Natural Language Processing is a subfield of Artificial Intelligence that focuses on the computational analysis and understanding of human language, particularly text and speech. Its primary goal is to transform unstructured linguistic data into structured, machine-readable representations, enabling computers to interact with human language in a meaningful and intelligent way. Using techniques from several disciplines, such as linguistics (to model language structure and grammar), statistics (to identify patterns and probabilistic relationships), and machine learning (to build systems that learn from data). In recent years, deep learning has significantly advanced the capabilities of NLP, allowing machines to understand and generate language with greater accuracy and fluency [28]. The following chapter will explore the role of deep learning in NLP in more detail.

As a result of these interdisciplinary advances, modern NLP systems can perform a wide variety of tasks. These include machine translation, sentiment analysis, information retrieval from large text corpora, and conversational interfaces such as chatbots and virtual assistants. All of these tasks rely on the system's ability to grasp not only the literal meaning of words but also their context and nuanced intent. Understanding such subtleties is essential for building NLP systems that interact effectively with human users.

2.1 Backstory

Although the origins of NLP date back to the 1960s, a major transformation began in the early 2000s with the advent of neural language models. A seminal contribution was made in the area of *Neural Language Modeling*, where the probability of the next token (words, character sets, or combinations of words) is predicted given the preceding n words. Bengio et al. [30] introduced a feedforward neural network along with a lookup table that represents these n previous words in sequence.

Subsequently, Collobert et al. [31] proposed the application of *Multitask Learning* in NLP, where convolutional models with max-pooling were used to simultaneously perform part-of-speech tagging and named entity recognition. Later, Mikolov et al. [32] introduced the concept of *Word Embeddings*, where dense vector representations of words replaced traditional sparse bag-of-words approaches, effectively capturing semantic relationships.

The emergence of word embeddings opened the door for more sophisticated neural models in NLP. One such development was the introduction of architectures capable of processing variable-length input sequences. Sutskever et al. [33] proposed the *Sequence-*

to-Sequence framework, which uses an encoder-decoder structure to map input sequences to output sequences via an intermediate vector representation.

Neural networks have since played a transformative role in the evolution of NLP. While their adoption was initially limited, by 2013, extensive research had demonstrated their potential across a range of NLP tasks. This shift led to the implementation of various neural architectures, including *Convolutional Neural Networks* (CNNs), which were originally developed for image classification but later adapted to text processing tasks.

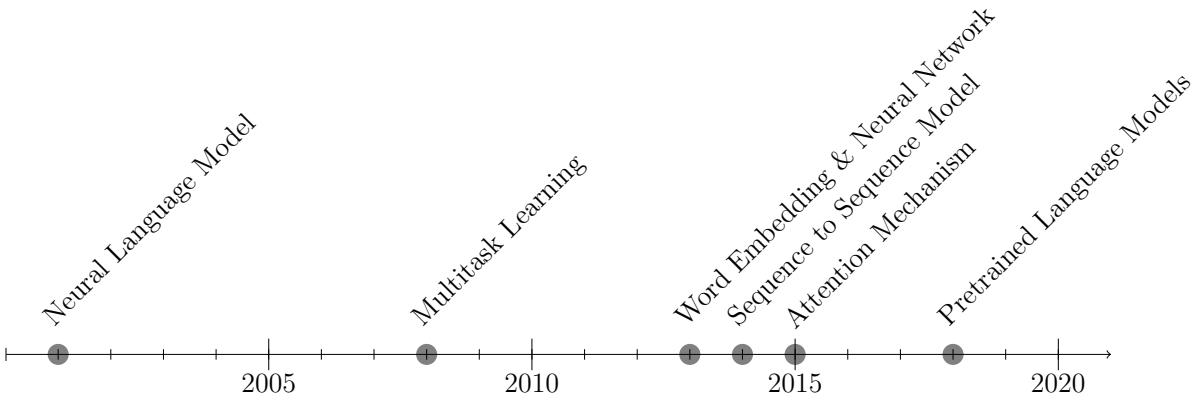

Figure 3: Key developments in the evolution of neural approaches to NLP (2000–2020)

Later, the use of CNNs became prominent in addressing various NLP tasks such as sentence classification [34], sentiment analysis [35], text classification [36], text summarization [37], machine translation [38], and question answering [39]. An article by Newatia (2019) [40] illustrates the general architecture of CNNs and their application in NLP, while Wang and Gang [41] provide a broader overview of CNN applications in the field.

Beyond convolutional models, another key advancement in NLP was the introduction of *Recurrent Neural Networks* (RNNs), which are particularly suited to sequential data due to their ability to maintain hidden states across time steps [42]. RNNs have proven to be effective for processing various forms of sequence data such as text, time series, speech, and video. A notable enhancement of the RNN architecture is the *Long Short-Term Memory* (LSTM) network [43], designed to selectively retain important information over longer sequences while discarding irrelevant data. Subsequently, a simplified variant known as the *Gated Recurrent Unit* (GRU) was introduced, offering comparable or superior performance to standard LSTMs in many tasks [44].

A major milestone in the evolution of deep learning for NLP came with the development of *attention mechanisms* [20], which allow models to dynamically focus on relevant parts of the input sequence. These mechanisms became fundamental to the success of the *Transformer* architecture [45], which effectively models long-range dependencies in text through self-attention. However, traditional Transformers are limited by a fixed-length context in language modeling tasks.

To address this limitation, Dai et al. [46] proposed the *Transformer-XL* (extra-long), an architecture capable of capturing dependencies beyond fixed-length segments. Further improvements were introduced in the *Compressive Transformer* by Rae et al. [47], which increases memory efficiency by compressing past hidden states for long-term learning. A comprehensive survey by Otter et al. [48] explores the role of deep learning in NLP and includes many relevant references for further reading.

Finally, the introduction of *Bidirectional Encoder Representations from Transformers* (BERT) [49] and its successors has had a profound impact on NLP, enabling models to leverage deep bidirectional context and significantly improve performance across a wide range of tasks.

2.2 Approaches

NLP is typically divided into two interrelated areas [50]. On the one hand, *Natural Language Understanding* (NLU) aims to enable machines to comprehend the meaning behind words, phrases, sentences, and discourse, considering contextual and semantic cues. On the other hand, *Natural Language Generation* (NLG) focuses on enabling machines to produce fluent, coherent, and contextually appropriate language, allowing them to communicate information in a human-like manner. However, before delving into these areas, it is essential to understand how machines interpret language at a fundamental level.

2.2.1 Language Interpretation

Language interpretation refers to the process through which machines convert unstructured human language into structured representations that can be understood and manipulated computationally. This process acts as a bridge between raw linguistic input and higher-level tasks such as understanding and generation.

The first step in this pipeline typically involves *tokenization*, where text is broken down into meaningful units such as words, subwords, or characters [51]. For example, consider the following sentence:

"The cat sat on the mat."

After tokenization, the sentence might be split into individual words:

`[the, cat, sat, on, the, mat]`

Each word is then converted into a unique numerical index using a vocabulary lookup. For instance:

`the` → 75, `cat` → 94, `sat` → 44, `on` → 86, `mat` → 18

This transformation can be visualized as a matrix where rows represent samples (e.g., different sentences), and columns represent word positions (see Figure 4).

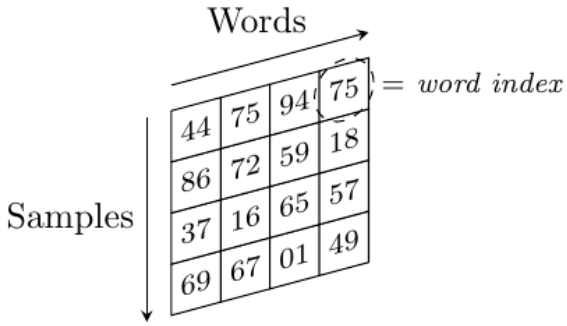

Figure 4: Tokenized input represented as word indices. Each number corresponds to a specific word in the vocabulary.

These word indices are then mapped into dense vector representations using *word embeddings*, such as Word2Vec [52], GloVe [53], or contextual embeddings from models like BERT [49]. This process captures semantic relationships between words by assigning similar vectors to semantically related terms. The result is a 3D tensor where each word is now associated with a fixed-dimensional real-valued embedding vector (see Figure 5).

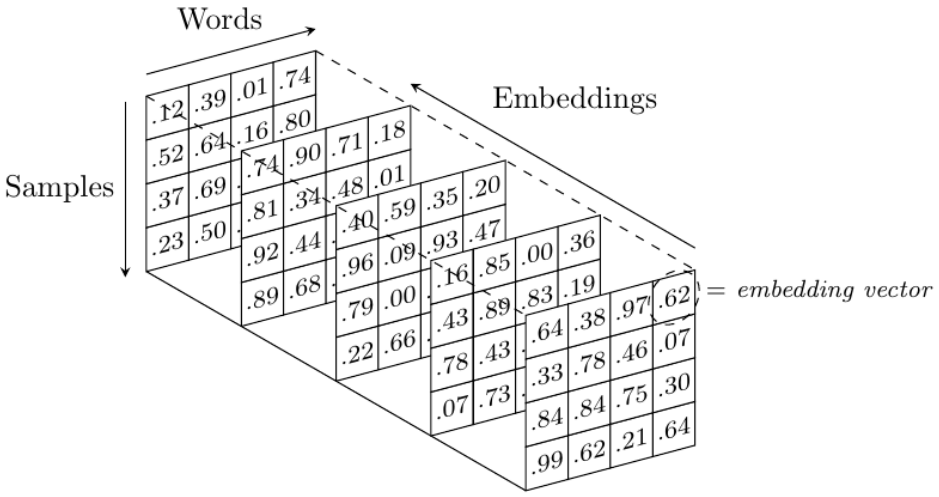

Figure 5: Word embeddings: each word index is mapped to a high-dimensional dense vector representing semantic information.

After obtaining a numerical representation of language through techniques like embeddings or one-hot encoding, further linguistic processing is typically applied. This may include: *Part-of-Speech* (POS) *tagging* [54], which assigns each word in a sentence its corresponding grammatical category (e.g., noun, verb, adjective); *Named Entity Recognition*

(NER) [55], which identifies and classifies named entities such as persons, organizations, or locations; and *Syntactic Parsing* [56], which analyzes the grammatical structure of a sentence to uncover hierarchical phrase relationships. These processing steps enrich the raw text with structural and functional annotations, aiding in the interpretation of grammatical dependencies and the roles that words play in the sentence.

2.2.2 Natural Language Understanding

Natural Language Understanding (NLU) refers to the ability of machines to interpret, process, and derive meaning from human language by recognizing relevant components such as entities, concepts, emotions, and user intent [50]. It underpins various intelligent applications such as chatbots, voice-controlled assistants, and automated customer service systems, where accurate interpretation of human input is critical.

Given that NLU is deeply rooted in the field of linguistics (which studies the structure, meaning, and context of language), understanding various levels and terminologies of linguistics is essential to developing systems that can process language with a human-like understanding. Below, the primary levels of linguistic analysis commonly integrated in NLP are described:

- **Phonology**: *Phonology* concerns the systematic organization of sounds in the spoken language [57]. Although its direct relevance to text-based NLP is limited, phonology is central in speech processing systems to map audio signals to phonemes and words.
- **Morphology**: *Morphology* studies the internal structure of words and the rules by which they are formed, analyzing components such as roots, prefixes, and suffixes [58]. Morphological processing supports tasks such as stemming, lemmatization, and POS tagging.
- **Lexical**: *Lexical analysis* focuses on the properties of individual words, including their parts of speech, meanings and relationships within a lexicon [59]. It forms the basis for many downstream NLP tasks, such as entity recognition and word sense disambiguation.
- **Syntactic**: *Syntax* governs the grammatical arrangement of words in a sentence. Syntactic analysis (or parsing) structures text into hierarchies such as noun phrases and verb phrases and reveals dependencies between words [18].
- **Semantic**: *Semantics* relates to the meaning of words, phrases, and sentences. Semantic analysis involves determining word senses, resolving ambiguity, and modeling conceptual relationships between terms [60].
- **Discourse**: *Discourse analysis* examines the relationships between sentences in a text or conversation. It enables tasks such as coreference resolution, coherence tracking, and dialog state management [59].

- **Pragmatic**: *Pragmatics* studies how context influences the interpretation of language, including the intention of the speaker, implied meanings, politeness strategies, and sarcasm [61]. It is essential for generating or interpreting responses in dialogue systems.

2.2.3 Natural Language Generation

Natural Language Generation (NLG) is the process of automatically producing coherent and contextually appropriate text, or speech, from non-linguistic internal representations such as structured data, knowledge bases, or logical forms. It is a fundamental component of NLP and is often considered the counterpart to NLU, focusing not on interpreting the language but on constructing it.

The NLG process typically unfolds in four main stages: (1) identifying communicative goals, (2) planning how those goals can be achieved given the context and available resources, (3) selecting and structuring content, and (4) realizing the output in natural language form. Below, the Natural Language Components are described.

- **Speaker and Generator**: To generate a text, it is necessary to have a speaker (or an application) and a generator (or a program) that renders the speaker's intentions into a fluent phrase relevant to the situation.
- **Components and Levels of Representation**: The process of language generation involves the following interweaves tasks:
  - *Content selection*: Information should be selected and included in the set. Depending on how this information is parsed into representational units, parts of the units may have to be removed, while some others may be added by default.
  - *Textual Organization*: The information must be textually organized according to the grammar, it must be ordered both sequentially and in terms of linguistic relations like modifications.
  - *Linguistic Resources*: To support the information's realization, linguistic resources must be chosen. In the end of these resources it will come down to the choice of particular words, idioms, syntactic constructions, etc.
  - *Realization*: The selected and organized resources must be realized as an actual text or voice input.
- **Application or Speaker**: This is only to maintain the model of the situation. The speaker just initiates the process and does not participate in the language generation. It stores the history, structures the content that is potentially relevant, and deploys a representation of what it knows. All these form the situation, while selecting a subset of propositions that the speaker has. The only requirement is that the speaker make sense of the situation.

2.3 Applications

NLP has a wide range of applications across numerous domains, including but not limited to machine translation, email spam detection, information extraction, summarization, and question answering. In the following sections, we describe some of the most impactful areas of application.

2.3.1 Machine Translation

With a significant portion of the world's population now online, ensuring that digital content is accessible in different languages has become a major challenge. One of the primary obstacles to achieving this accessibility is the language barrier. Human languages differ widely in terms of vocabulary, grammar, syntax, and semantics, making translation a complex task.

Machine Translation (MT) refers to the automatic translation of text or speech from one language to another, traditionally using statistical or rule-based models. A well-known example of a statistical machine translation engine is *Google Translate*.

The key difficulty in MT lies not in translating words directly but in preserving the intended meaning, grammatical structure, and temporal aspects (e.g., tense, aspect) of the original sentence. *Statistical Machine Translation* (SMT) systems rely on large volumes of bilingual (parallel) corpora to identify probabilistic correspondences between expressions in two languages. These systems use statistical methods to compute the most likely translation of a given phrase based on observed patterns.

In September 2016, Google introduced a major breakthrough by launching a neural machine translation system based on deep learning and artificial neural networks [62]. Unlike traditional SMT approaches, *Neural Machine Translation* (NMT) models are capable of capturing long-range dependencies and generating more fluent and contextually accurate translations.

Over the years, several automated evaluation metrics have been proposed to assess the quality of machine translations by comparing system-generated outputs (hypotheses) with reference translations. Notable metrics include *Word Error Rate* (WER) and *Position-Independent Word Error Rate* (PER) [63], Generation String Accuracy [64], Multi-Reference Word Error Rate [65], the *Bilingual Evaluation Understudy* (BLEU) score [66], and the NIST score [67]. These metrics aim to approximate human judgment of translation quality and often correlate strongly with human evaluations of fluency and adequacy [66].

2.3.2 Text Categorization

Text categorization, also known as text classification, involves assigning predefined categories or labels to incoming textual data based on its content. These systems are designed to process large volumes of unstructured information, such as official documents, military reports, market data, or news articles, and automatically organize them into structured categories or indices.

An illustrative example is the Construe system developed by the Carnegie Group [68], which classifies Reuters newswire articles. By automating the categorization process, such systems significantly reduce the manual effort required from human indexers, leading to increased efficiency and scalability.

Text categorization has been widely adopted in industries. For example, customer support systems often use it to classify incoming complaint tickets or service requests, automatically routing them to the appropriate department or handling agent. This automation improves response time, ensures accurate prioritization, and improves overall service delivery.

2.3.3 Spam Filtering

Spam filtering plays a critical role as the first line of defense against unwanted and potentially harmful emails. One of the core challenges in spam detection lies in minimizing false positives (legitimate emails marked as spam) and false negatives (spam emails that bypass the filter). This problem exemplifies the broader challenge of extracting semantic meaning from textual data.

There are several types of spam filters, each utilizing different strategies to detect unsolicited messages:

- **Content Filters**: Analyze the actual content of the email to identify keywords, phrases, or patterns typically associated with spam.
- **Header Filters**: Examine the metadata and header information of an email (e.g., sender address, routing details) to detect spoofed or suspicious data.
- **Blacklist Filters**: Block all emails originating from addresses or domains listed in a predefined blacklist.
- **Rule-Based Filters**: Apply user-defined or system-defined rules, such as blocking emails from specific senders or those containing certain keywords.
- **Permission Filters**: Restrict message delivery to senders who have been explicitly authorized by the recipient.
- **Challenge-Response Filters**: Require senders to complete a verification task (e.g., entering a code) before their message is accepted by the system.

Spam filtering is fundamentally a text categorization problem. In recent years, numerous machine learning approaches have been employed to enhance anti-spam systems, including Rule Learning [69], Naïve Bayes [70], Memory Based Learning [71], *Support Vector Machines* (SVM) [72], Decision Trees [73], Maximum Entropy Model [74], Hash Forest and a Rule Encoding Method [75]. All of these techniques allow for improved classification accuracy by learning from labeled datasets and adapting to evolving patterns of spam behavior.

2.3.4 Information Extraction

Information Extraction (IE) focuses on identifying and extracting relevant phrases or data points from unstructured textual content. The information retrieved can serve a wide range of applications, including summarization, keyword identification, database population, and the classification of documents into predefined categories.

A well-known example is the CONSTRUE system, originally developed for Reuters, which was employed to classify news articles efficiently and reduce the manual workload of human indexers [68].

While many IE systems are effective at extracting individual terms or entities from documents, identifying the relationships between these elements remains a significant challenge. For instance, the PROMETHEE system was designed to extract lexico-syntactic patterns that correspond to specific conceptual relationships, thus improving relation extraction capabilities [76].

Effective IE systems typically operate on multiple levels of linguistic processing. These range from low-level tasks such as tokenization and word recognition to higher-level analysis like discourse understanding at the document scale. One practical implementation is the *Blank Slate Language Processor* (BSLP), which has been applied to analyze real-world natural language data, such as responses to open-ended questionnaires in the field of advertising [77].

2.3.5 Summarization

In the digital era, information overload has become a pressing issue, as our access to knowledge and data far exceeds our cognitive capacity to process and interpret it. This imbalance continues to grow, emphasizing the urgent need for effective summarization techniques that can distill essential information while preserving its core meaning. Summarization not only aids in identifying the most relevant content within vast datasets but also enables the extraction of underlying emotional or contextual nuances.

Text summarization can be broadly classified based on the number of documents involved: *single-document summarization* and *multi-document summarization* [78]. Several techniques have been proposed to address these tasks, including:

- *Bayesian Sentence-Based Topic Model* (BSTM): This model leverages both term-sentence and term-document associations to generate summaries from multiple documents. It emphasizes identifying common themes across texts to construct a coherent summary [79].
- *Factorization with Given Bases* (FGB): In this approach, sentence bases are predefined and used within a language model framework. It utilizes document-term and sentence-term matrices to simultaneously group and summarize documents [80].
- *Topic Aspect-Oriented Summarization* (TAOS): TAOS organizes summarization around topic factors (features that describe specific topics, such as capitalized words representing named entities). It accounts for multiple aspects within a topic and adapts feature preferences accordingly to generate focused summaries [81].

2.3.6 Dialogue Systems

Dialogue systems are increasingly prevalent in real-world applications, ranging from customer support services to task-oriented automation. In support-oriented dialogue systems, contextual awareness is essential to maintain coherent and meaningful interactions. In contrast, task-oriented systems designed to perform specific actions often require less contextual processing. These systems operate across various linguistic levels, including phonemic and lexical, to interpret and generate human-like responses.

Advanced, habitable dialogue systems strive for fully automated natural language interactions by incorporating all levels of linguistic analysis [82]. This development enables more natural and human-like conversations between users and machines. Examples of such systems include widely used virtual assistants such as *Google Assistant*, *Microsoft Cortana*, *Apple Siri*, and *Amazon Alexa*, which exemplify the integration of NLU and NLG to facilitate intuitive human-computer interaction.

2.3.7 Medicine

Natural Language Processing has also found impactful applications in the field of medicine. One of the most significant large-scale efforts is the *Linguistic String Project–Medical Language Processor* (LSP-MLP), which facilitates the extraction and summarization of medical information from clinical texts. The system assists healthcare professionals by identifying signs and symptoms, drug dosages, and response data, with the aim of detecting potential side effects and highlighting critical data points [83, 84].

The National Library of Medicine is developing a system called *The Specialist System*, designed to serve as an Information Extraction tool specifically for biomedical knowledge bases, such as Medline abstracts [85].

Furthermore, Columbia University has created the *MEDical Language Extraction and Encoding System* (MEDLEE), which is capable of processing narrative clinical reports.

MEDLEE identifies relevant clinical information and converts unstructured text into a structured format suitable for analysis and integration into electronic health records [86].

2.4 Challenges

Despite significant progress in NLP in recent years, the field continues to face a wide range of challenges. These challenges are inherent in the complexities of human language and pose difficulties in designing robust, context-aware, and generalizable models. Below, some of the most important issues are outlined [50]:

- **Contextual Understanding and Synonymy**: A primary challenge in NLP is the interpretation of contextual words and phrases. The same word or phrase can have different meanings depending on the context, which humans can effortlessly interpret but machines struggle to understand. This issue is further complicated by synonymy, the use of multiple words to express the same idea. For example, terms such as *large*, *huge*, and *big* might be used interchangeably by different speakers, requiring models to recognize and process semantic equivalence accurately.
- **Homonyms and Speech Ambiguity**: Homonyms (words that are pronounced the same but have different meanings) pose another significant obstacle, especially for speech-to-text and question-answer systems. Since homonyms are not differentiated in spoken form, interpreting their intended meaning without additional context becomes a major difficulty for NLP systems.
- **Sarcasm, Irony, and Ambiguity**: Understanding figurative language such as sarcasm and irony remains a formidable task in NLP. Sentences intended sarcastically may convey meanings opposite to their literal interpretations, necessitating models that can infer deeper emotional and contextual cues. Similarly, linguistic ambiguity (where a sentence can be interpreted in more than one way) demands sophisticated disambiguation techniques to ensure accurate understanding.
- **Informal Language and Cultural Specificity**: Languages are replete with informal phrases, idioms, and culturally specific expressions, making it difficult to build generalized models for broad applications. Although training in large and diverse datasets can mitigate some issues, capturing the nuances of region-specific language remains a major challenge. Words and expressions often carry different meanings across geographical regions and cultural contexts, further complicating the generalization of the model.
- **Domain-Specific Language Use**: The same terms can carry different meanings across various domains such as education, healthcare, law, and defense. NLP models trained in one domain often fail to generalize to another without domain-specific adaptation. Developing systems that can handle multiple domains effectively without losing performance is an open problem.

- **Spelling Errors and Writer Intentions**: Misspelled words add another layer of complexity. Although modern grammar correction systems have improved significantly, understanding the true intent of the writer (particularly when affected by informal language, sarcasm, or regional expressions) remains a difficult task.
- **Language Inclusivity and Generalization**: Although NLP systems for major languages have made great strides, there is still a lack of high-performing models for low-resource languages and users with varying levels of linguistic knowledge. Ensuring inclusivity and accessibility in NLP across diverse user groups and linguistic backgrounds remains an important research objective.

3 Fundamentals of Large Language Models

In this chapter, we explore the core concepts that underlie LLMs. We begin by reviewing their historical development and potential directions for future research. Then, we examine their main applications, tokenization methods, architectural designs, types of LLM, benchmarks, and LLM drawbacks. These foundational concepts are essential for the practical component of this work, in which we apply LLMs to classify social media posts.

3.1 History

To understand the evolution of Language Models, we refer to the comprehensive survey by Wang et al. [87], which identifies four major phases in their development. Figure 6 illustrates this progression: beginning with Statistical Language Models (SLMs) in the 1990s, followed by Natural Language Models (NLMs), which dominated until 2017. This was succeeded by the emergence of Pre-trained Language Models (PLMs), and ultimately by the rise of Large Language Models (LLMs) around 2020, which represent the state-of-the-art today.

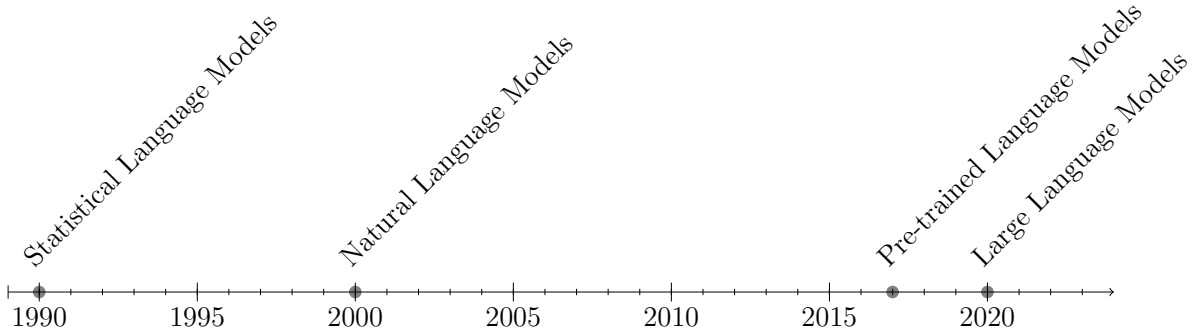

Figure 6: Evolution of Language Model paradigms (1990–2020)

3.1.1 Statistical Language Models

SLMs emerged in the 1990s as a way to represent natural language using mathematical and probabilistic tools. Their main purpose is to model the likelihood of word sequences by capturing contextual information from language in a statistical manner. In essence, the core idea behind an SLM is to estimate the probability of a sentence occurring in a given language, based on patterns observed in large text corpora [88].

To illustrate this idea, consider the sentence "I am very happy". In this case, each word in the sentence is assigned an index: w_1 corresponds to "I", w_2 to "am", w_3 to "very", and w_4 to "happy". The goal is to determine the probability of the entire sentence, denoted as $P(S)$. This is expressed as:

$$P(S) = P(w_1, w_2, w_3, w_4) = P(I, am, very, happy)$$

To compute this probability, we apply the chain rule of probability, which allows us to decompose the joint probability of the sentence into a sequence of conditional probabilities:

$$P(I, am, very, happy) = P(I) \cdot P(am \mid I) \cdot P(very \mid I, am) \cdot P(happy \mid I, am, very)$$

In this formulation, each term represents the probability of a word given the words that precede it. For example, $P(I)$ is the probability of the word "I" appearing, while $P(am \mid I)$ is the probability that "am" follows "I". When we multiply $P(am \mid I)$ by $P(I)$, we obtain the probability that the phrase "I am" appears in the text, which is written as $P(I, am) = P(I) \cdot P(am \mid I)$.

To estimate these probabilities in practice, we rely on a technique called Maximum Likelihood Estimation (MLE). This method allows us to approximate the probability of a word given its context by using frequency counts from a large dataset. If the dataset is large enough, relative frequencies can serve as good estimates of actual probabilities. MLE is expressed as follows:

$$P(w_i \mid w_1 w_2 \dots w_{i-1}) = \frac{P(w_1 \dots w_i)}{P(w_1 w_2 \dots w_{i-1})} = \frac{C(w_1 w_2 \dots w_i)}{C(w_1 w_2 \dots w_{i-1})}$$

Here, $C(\cdot)$ denotes the number of times a given word sequence appears in the training data. This formula enables us to calculate the probability of a word w_i appearing after the specific sequence of words w_1 through w_{i-1}. Once these probabilities are computed, we can predict the most likely next word by selecting the one with the highest probability given the current context.

This approach assumes that the likelihood of each word depends on the previous $n-1$ words, a simplification known as the n-gram model. Using n-grams makes the model computationally feasible, as it reduces the number of dependencies to consider. However, to compute conditional probabilities efficiently, it is necessary to precompute and store the counts $C(X)$ for all relevant word sequences X of length n.

One of the main challenges of n-gram models is that the number of possible word sequences grows exponentially with the size of the vocabulary. For example, with a vocabulary of 1000 words, there are 1000^n possible combinations for sequences of length n. This makes it difficult to store and compute probabilities for all possible n-grams when n becomes large. As a result, in practice, n is usually limited to small values such as 2 or 3. This means that each word is assumed to depend only on the previous one or two words, which makes the model more manageable but also limits its ability to capture long-range dependencies in language, leading to reduced accuracy in some cases.

3.1.2 Natural Language Models

NLMs make use of neural networks to estimate the probability of the next word in a sequence. This approach improves on SLMs, as NLMs are better equipped to handle longer sequences and alleviate the constraints introduced by small values of n in traditional n-gram models. Before diving into the structure of neural networks, it is helpful to understand how words can be represented numerically through word vectors.

Although humans easily grasp the meaning and relationships between words (for example, recognizing that "cat" and "dog" are more semantically related than "cat" and "tree") computers do not have this intuitive understanding. Since computers process information in binary form, words must be converted into numerical formats that preserve semantic relationships. This is achieved using word vectors, which are fixed-length numerical representations of words. Each word is mapped to a vector in a continuous space, and relationships between words can be understood through the geometric properties of these vectors. For instance, the angle between the vectors representing "cat" and "dog" is typically smaller than the angle between "cat" and "tree", reflecting their closer semantic connection.

One of the most well-known methods for generating word vectors is Word2Vec [52]. This model transforms words into dense vector representations in such a way that words with similar meanings end up close together in the vector space. Word2Vec's success is largely attributed to its foundation in neural networks, which mimic certain aspects of the human brain by using layers of interconnected units called neurons.

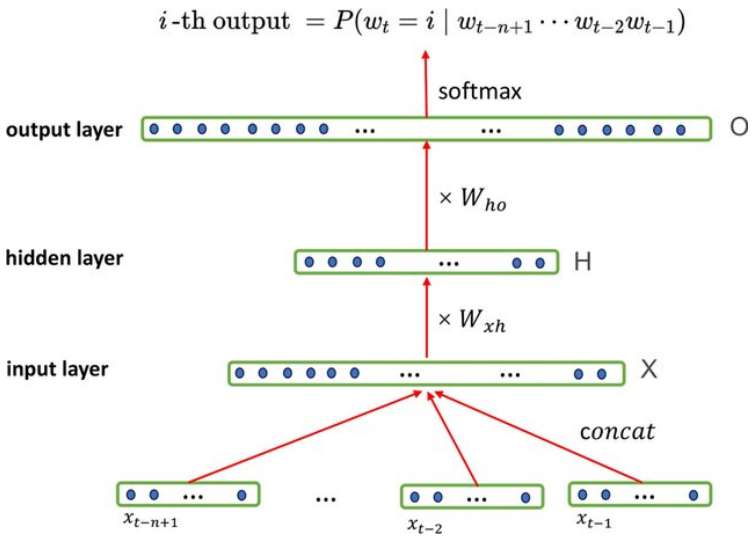

Figure 7: Neural Language Model

Figure 7 illustrates the basic structure of a neural language model, which consists of an input layer, one or more hidden layers, and an output layer. In this setup, each input word vector is denoted as $x_t \in \mathbb{R}^m$, where m represents the dimensionality of the word embeddings. These vectors are concatenated to form a single input matrix $X \in \mathbb{R}^{m \times (n-1)}$,

representing the context of the previous $n-1$ words. This input is then transformed by multiplying it with the weight matrix $W_{xh} \in \mathbb{R}^{m(n-1)\times h}$, which maps the input to a hidden representation. The resulting product is passed through a non-linear activation function, such as the sigmoid or tanh, to produce the hidden layer output H.

The hidden representation H is then used to predict the next word in the sequence. This is done by multiplying H with another weight matrix $W_{ho} \in \mathbb{R}^{h\times|V|}$, where $|V|$ is the size of the vocabulary, and h is the number of neurons in the hidden layer. The result is an output vector $O \in \mathbb{R}^{|V|}$, where each component corresponds to a score for a word in the vocabulary. To convert these scores into probabilities, the Softmax function is applied to the output vector, yielding a normalized probability distribution O' over the vocabulary.

The Softmax function for the i-th output is defined as:

$$O'_i = Softmax(O'_i) = \frac{exp(O_i)}{\sum_{j=1}^{|V|} exp(O_j)}$$

where O'_i is the probability assigned to the i-th word, and $|V|$ denotes the total number of words in the vocabulary. The Softmax function ensures that the output values form a valid probability distribution, where each probability lies in the range [0, 1] and the total sum of probabilities is 1. This final distribution allows the model to choose the most likely next word in the sequence, based on the context provided by the previous words.

3.1.3 Pre-trained Language Models

PLMs are initially trained on large volumes of unlabeled text data. This initial phase, known as pre-training, allows the model to learn the basic structure of language, including vocabulary, syntax, semantics, and logical patterns. Once this foundational knowledge is acquired, the same model can be adapted to perform a wide range of downstream NLP tasks, such as machine translation, text summarization, and question answering.

To tailor the model to a specific task, a second training phase is typically required. This process, referred to as fine-tuning, involves training the model on a smaller task-specific dataset. The combination of pre-training and fine-tuning constitutes a widely used learning paradigm that has proven effective in adapting general-purpose models to specialized applications.

A useful way to conceptualize this paradigm is through an analogy drawn from martial arts fiction. In such narratives, a person aspiring to become a martial arts expert first develops a strong internal foundation by practicing a wide variety of techniques. This foundational training is analogous to the pre-training phase. Later, the individual learns a specific combat technique, such as the use of a sword or a palm strike, and masters it more efficiently relying on the principles learned during the foundational training. This final stage corresponds to the fine-tuning process in machine learning.

Numerous influential studies on PLMs have adopted this pre-training and fine-tuning approach, introducing various architectures that follow this methodology. Notable examples include GPT-2 [89] and BERT [49], which have demonstrated strong performance in a wide range of NLP tasks leveraging the benefits of this two-phase training strategy.

3.1.4 Large Language Models

LLMs are trained on extensive text corpora and contain an exceptionally high number of parameters, often reaching tens or even hundreds of billions. Notable examples include GPT-3 [25], GPT-4 [90], PaLM [91], and LLaMA [92].

The primary objective of LLMs is to equip machines with the ability to comprehend human instructions and respond in ways that align with human values. Unlike traditional models that may require adaptation to specific tasks or domains, LLMs typically follow a two-stage process: an initial pre-training phase in a vast general purpose dataset, followed by a refinement phase aimed at aligning the model's behavior with human intentions and ethical considerations. This alignment stage does not involve domain adaptation, but rather focuses on fine-tuning the model to better understand and reflect human preferences.

Compared to earlier PLMs, LLMs demonstrate a higher degree of flexibility and generalization. They are capable of performing well across a broad spectrum of tasks without the need for extensive task-specific training. This is largely due to the dramatic increase in model size, the scale of the training datasets, and the computational power used during training. These factors have led to significant improvements in performance and have revealed emerging abilities that are typically not present in smaller models.

A central concept in understanding LLMs is the notion of parameters. Parameters are the internal values within a neural network that are learned during training. They determine how the input data are transformed at each layer of the model to produce an output. In the context of language models, parameters control how word sequences are processed, how patterns are recognized, and how the model generates text. The more parameters a model has, the more complex patterns it can potentially capture. Therefore, a model with a larger number of parameters generally has a higher capacity to learn from data and generalize to new tasks.

For example, GPT-3 introduces a major leap in capability over GPT-2, one key reason being its ability to leverage in-context learning. This means that GPT-3 can interpret and use the examples provided in the prompt to improve its responses without needing additional training. GPT-2 lacks this capacity, illustrating the importance of model scale and architecture in determining performance. To give a concrete comparison, the largest version of GPT-2 includes approximately 1.5 billion parameters and was trained on 40 GB of text. In contrast, the largest version of GPT-3 comprises 175 billion parameters and was trained on 570 GB of text data.

3.2 Tokenization

As introduced in the previous chapter on NLP, *tokenization* refers to the process of dividing sequences of symbols, typically natural language text, into smaller discrete units known as *tokens* [93]. These tokens serve as the fundamental building blocks on which LLMs operate. In this section, we explore the concept of tokenization in greater depth and illustrate how it enables LLMs to process input text and generate coherent, context-aware responses that appear "intelligent" to users. While the underlying model architectures will be discussed in the next section, it is essential first to understand how textual data is transformed before it is ingested by the model, and how the model's output is converted back into natural language.

3.2.1 Backstory

The tokenization of linguistic data has long been a foundational practice in the field of NLP [94]. Traditionally used to segment text into words or morphemes, tokenization gained renewed significance with the emergence of deep neural models, where the granularity and representation of linguistic units became crucial to model performance.

A major change in neural NLP occurred with the introduction and widespread adoption of *Byte-Pair Encoding* (BPE) [95], which established subword tokenization as the preferred strategy for modern applications. Originally adapted from a data compression algorithm, BPE was designed to address one of the main challenges in neural machine translation: the presence of *out-of-vocabulary* (OOV) terms. OOV refers to tokens that do not appear in the model's fixed vocabulary and therefore cannot be processed directly. Subword tokenization methods such as BPE alleviate this issue by decomposing rare or unseen words into smaller, known units, thus enabling open-vocabulary modeling while still preserving meaningful subword information.

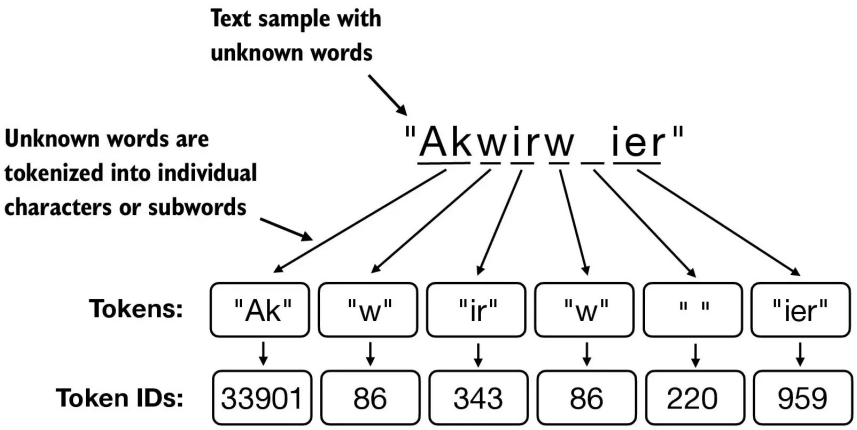

Figure 8: Byte-Pair Encoding example.

In essence, BPE begins with a vocabulary consisting only of individual characters and iteratively merges the most frequent adjacent pairs of symbols into new tokens. This procedure continues until a predefined vocabulary size is reached, producing a representation that balances the granularity of character-level and word-level tokenization. Figure 8 illustrates this process, in which words are first divided into characters or subword units, which are then merged and assigned unique token identifiers.

BPE quickly replaced earlier heuristic and rule-based tokenizers such as Morfessor [96] and Moses [97], which were commonly used in statistical NLP pipelines. It also paved the way for more advanced data-driven tokenization models, including WordPiece [98] and Unigram [99], which have since become among the most widely adopted approaches in modern NLP systems.

3.2.2 Text Interpretation

The process of *text interpretation* involves transforming a sequence of symbols (typically characters of an input alphabet) into a corresponding sequence of tokens drawn from a predefined vocabulary. This transformation is a fundamental step that enables language models to operate on linguistically meaningful and statistically manageable components of language.

Importantly, LLMs are inherently multilingual, not in the sense that they understand or distinguish languages in the human sense, but because they process all input uniformly as token sequences. Rather than identifying and parsing input based on specific languages, these models learn statistical patterns on tokens [100]. The behavior of an LLM is thus entirely shaped by the distribution of tokens in its training data. The more diverse and extensive the training corpus, the wider the range of tokens and linguistic structures the model can encode. The apparent "intelligence" of the model emerges from the relationships between these tokens, captured through the learned parameters within the model architecture.

Once a sequence is tokenized, each token is mapped to a unique identifier (token ID), which is the actual input passed into the neural architecture. These token IDs serve as reference points for the embedding layers, attention mechanisms, and all subsequent computations performed by the model.

A practical demonstration of this process is available through the OpenAI Tokenizer tool[2], which provides real-time visualization of how input text is tokenized. For example, as illustrated in Figure 9, a sample input of 252 characters is segmented into 53 tokens.

[2] https://platform.openai.com/tokenizer

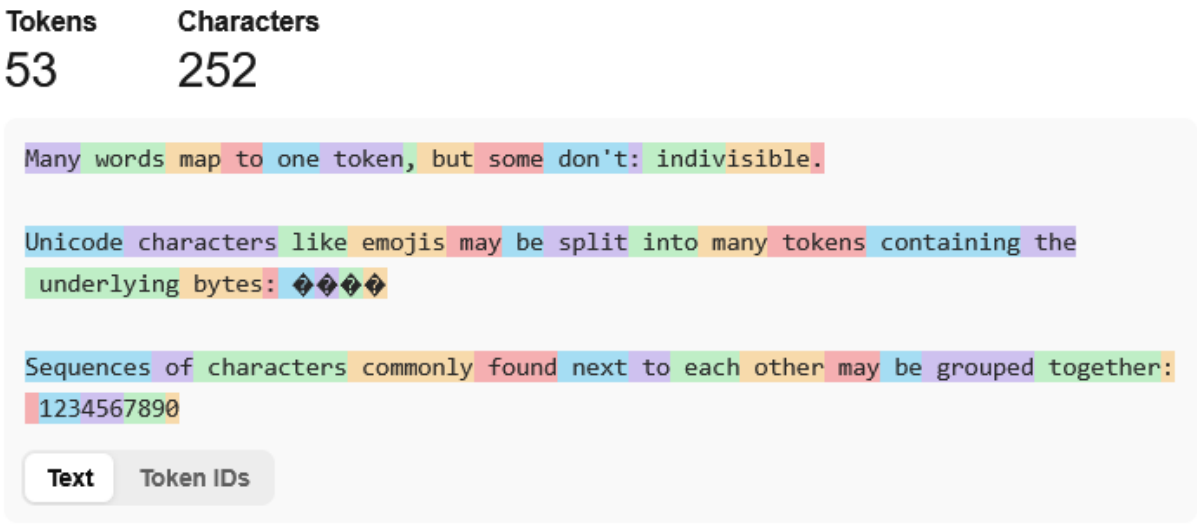

Figure 9: Conversion of raw text into tokens using OpenAI's Tokenizer

Each of these tokens is then assigned to a unique identifier, as shown in Figure 10, which illustrates the second step of the tokenization process, the conversion from token strings to numerical token IDs.

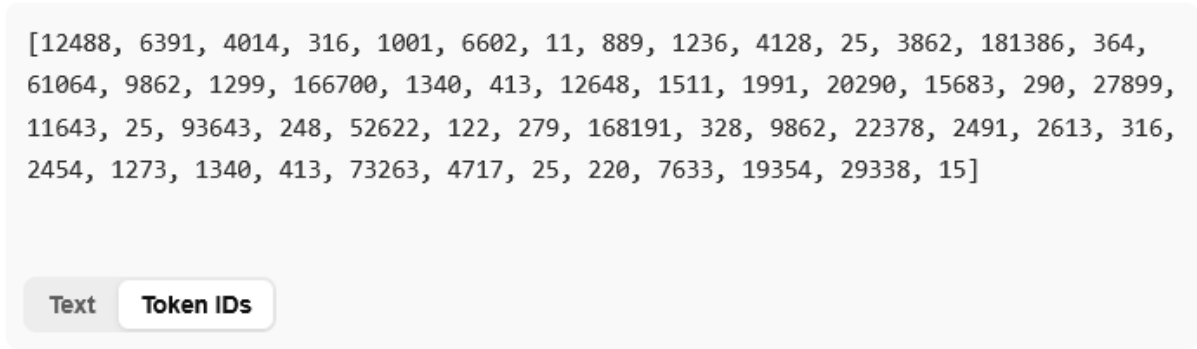

Figure 10: Mapping of tokens to their corresponding token IDs

The tokenization illustrated corresponds to the GPT-4o and GPT-4o mini models. However, the OpenAI Platform allows users to switch between different models. Changing the selected model will show how each model segments the text differently, highlighting that tokenization strategies vary between architectures, depending on the vocabulary and algorithm used during pretraining.

3.2.3 Drawbacks

Despite its foundational role, tokenization introduces several challenges. One major issue is *ambiguity* [101], where multiple token sequences may correspond to the same original

text, especially when tokenizers are non-deterministic or support multiple valid segmentations. Another issue is *inconsistency* [93], which arises when the tokenization process leads to discrepancies between the statistical distributions of the original text and its tokenized representation. Such inconsistencies can impact model performance, particularly in tasks that require precise alignment between inputs and outputs.

These problems are often exacerbated by preprocessing steps such as lowercasing, punctuation normalization, or the use of placeholder tokens. Moreover, from a computational point of view, efficient tokenization is critical. Ideally, tokenization should be linear in complexity with respect to the input length. However, ambiguity and decoding multiplicity can increase computational costs, particularly in decoding tasks that require recovering the original text or exploring multiple segmentation paths.

As LLMs continue to evolve and scale, optimizing tokenization for both performance and fairness remains an open research area, particularly in multilingual and morphologically rich languages where segmentation is more complex.

3.3 Architecture

This section aims to clarify the foundational principles and taxonomy underlying LLMs, with the goal of providing a deeper understanding of how these models work. By exploring their core components and mechanisms, the discussion aims to support informed decision-making when selecting the most suitable LLM for specific applications.

At the heart of LLMs lies the use of deep neural networks capable of capturing complex statistical patterns and semantic structures in language. This ability enables models not only to understand but also to generate coherent and contextually relevant text. Among the most prominent architectures is the GPT series, which leverages the transformer architecture [45] utilizes autoregression to predict the preceding tokens.

In this thesis, we focus on GPT-3 [25] as a representative model to illustrate the core principles of the LLM architecture. As shown in Figure 11, the structure of GPT-3 is depicted with emphasis on its modular design. For clarity, the illustration displays only two transformer layers, while the full model consists of 12 layers.

The following subsections will describe this architecture in a bottom-up manner, beginning with the input embedding layer, progressing through the transformer blocks, and finishing in the output layer following the Softmax activation. This step-by-step explanation will serve to demystify the internal workings of GPT models and highlight the architectural decisions that underpin their performance.

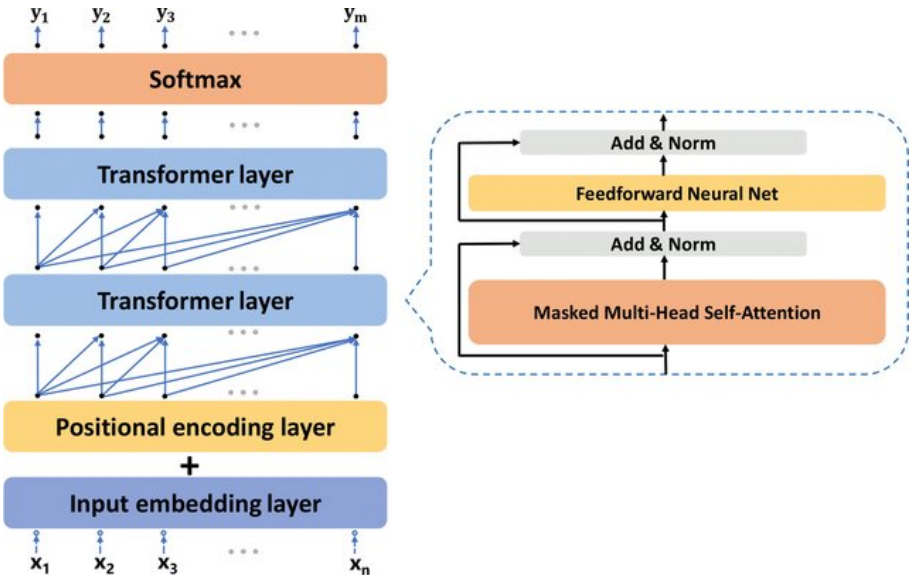

Figure 11: The GPT-3 model architecture

3.3.1 Input Embedding Layer

The first component of the GPT model architecture is the *input embedding layer*, which processes the tokenized input and converts it into numerical vectors suitable for the neural network. The model receives a sequence of token representations $(x_1, ..., x_n)$, where each x_i corresponds to a token obtained through the tokenization process described in the previous section. In the case of GPT-3, the maximum input sequence length is 2048 tokens.

Since machine learning models operate exclusively on numerical inputs, each token must be encoded as a vector. The first step is the construction of a fixed vocabulary, which in GPT-3 contains 50,257 unique tokens. Each token is initially represented as a one-hot vector of dimension 50,257, that is, a sparse binary vector in which only one element is set to 1 (indicating the index of the token in the vocabulary) and the rest are zeros. For example, the token "a" might be represented as $[1, 0, 0, ..., 0]$, while "an" would be $[0, 1, 0, ..., 0]$.

Given an input sequence of up to 2048 tokens, this encoding results in a matrix $I_E \in \mathbb{R}^{2048 \times 50,257}$, where each row corresponds to a one-hot vector representation of one token. However, these vectors are extremely sparse and inefficient in both storage and computation. To overcome this, the one-hot vectors are projected into a dense, lower-dimensional space using an embedding matrix $W_E \in \mathbb{R}^{50,257 \times d}$, where d is the embedding dimension. For GPT-3, d is set to 12,288 in the largest model configuration.

The transformation is performed through a matrix multiplication:

$$X_{\text{WordEmbedding}} = I_E \times W_E$$

resulting in an embedded input matrix:

$$X_{\text{WordEmbedding}} \in \mathbb{R}^{2048 \times 12{,}288}$$

3.3.2 Positional Encoding Layer

The position and order of words in a sentence play a crucial role in shaping its meaning. For example, consider the sentences: "*He is a good person and does not do bad things*" versus "*He is a bad person and does not do good things*". Although both contain similar words, a slight change in word order results in an entirely different interpretation. This illustrates why sequence information must be preserved when processing language.

In the GPT architecture, the *positional encoding layer* complements the input embedding layer by providing the model with information about the order of tokens in the sequence. Since transformer architectures process input tokens in parallel and lack inherent sequential awareness, positional encodings are essential to help the model understand the structure of the input.

GPT-3 implements positional encoding using sinusoidal functions, as originally proposed in the transformer architecture. The positional encoding for a token at position pos and dimension i is computed as:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{1}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2}$$

Here, pos is the token's position in the sequence, i refers to the dimension index, and d_{model} is the dimensionality of the embedding space (12,288 in the case of GPT-3). This alternating pattern of sine and cosine functions allows the model to uniquely encode each position in a way that generalizes well across varying sequence lengths.

But why use such complex functions for positional encoding? To understand this, it helps to first consider simpler approaches. The most basic method is to assign each token a scalar position value based on its order in the sequence, for example $[0, 1, 2, ..., T-1]$ for a sequence of length T. To align this with the dimension of the model embedding, one naive approach might be to expand each scalar position into a vector by repeating the same value across all dimensions. For instance, the positional encoding for the second word (position 1) could be represented as a vector of ones (e.g., $[1, 1, ..., 1]$), where the length of the vector matches the embedding dimension. However, this strategy introduces significant limitations, as it fails to differentiate between embedding dimensions and leads to poor generalization in longer sequences.

However, this method introduces a major issue: in longer sequences, later words receive disproportionately larger values, which can lead to numerical bias when combined with word embeddings. A common fix is to normalize these values by dividing by T, producing values in the range $[0, 1]$. Although this normalization keeps values bounded, it introduces inconsistency, given that the positional differences between adjacent tokens depend on the overall length of the sequence. As shown in Figure 12, in one sentence, the positional difference between neighboring tokens might be 0.5, while in another longer sentence, it could be just 0.125. These inconsistencies reduce the model's ability to generalize across varying input lengths.

Figure 12: Inconsistent position encoding using basic counting or normalization

To resolve this, Google introduced a trigonometric encoding scheme based on sine and cosine functions in the original transformer paper. This method generates a smooth, continuous, and length-independent encoding of position, which is particularly well suited for self-attention mechanisms [19] and allows the model to generalize across sequences of varying lengths.

The result is a positional encoding matrix $X_{\text{PositionalEncoding}}$, which is added element-wise to the word embedding matrix to form the final input to the transformer layers:

$$X = X_{\text{WordEmbedding}} + X_{\text{PositionalEncoding}}$$

where:

$$X \in \mathbb{R}^{2048 \times 12{,}288}$$

This combined input matrix retains both the semantic content (from the embeddings) and the sequential structure (from the positional encodings), enabling the transformer layers to process the text meaningfully.

3.3.3 Transformer Layer

The hidden layers of GPT-3 are built upon the transformer architecture, which consists of three main components: Masked Multi-Head Self-Attention, Feedforward Neural Networks, and Add & Norm layers. These components are shown on the right side of Figure 11 and are repeated in stacked transformer blocks throughout the model.

Masked Multi-Head Self-Attention

Masked Multi-Head Self-Attention combines two core ideas: self-attention and masking. To fully understand this mechanism, it is helpful to begin with the concept of self-attention.

In human cognition, attention allows us to focus on the most relevant parts of complex input. Self-attention in neural networks is inspired by this ability: it enables the model to determine which tokens in the input sequence are most relevant to the prediction of each token in the output sequence. For example, when predicting the word "happy" from the input "I am very", the model may assign attention weights of 0.7, 0.1, and 0.2 to the words "I", "am", and "very", respectively, reflecting their relative importance in predicting "happy". This mechanism involves the following steps:

- **Step 1:** The input matrix $X \in \mathbb{R}^{2048\times12,288}$ is projected into three separate matrices: query (Q), key (K), and value (V), by multiplying it with three learned weight matrices: $W_q \in \mathbb{R}^{12,288\times128}$, $W_k \in \mathbb{R}^{12,288\times128}$, and $W_v \in \mathbb{R}^{12,288\times128}$.
- **Step 2:** The attention scores are computed using the dot product of Q and $K^\top$, scaled, and passed through a Softmax function to obtain attention weights. These weights are then used to compute a weighted sum of the value matrix V:

$$Y = \text{Softmax}(QK^\top) \cdot V \tag{3}$$

  where $Q, K, V \in \mathbb{R}^{2048\times128}$ and $Y \in \mathbb{R}^{2048\times128}$.

This process can be intuitively understood with a real-world analogy. Imagine using a music search engine: Q is your query (for example, the song name), K is the database of all indexed song names and V is the corresponding audio files. The result Y is the audio track that matches your query.

While standard self-attention allows the model to consider all positions in the input sequence, masked self-attention introduces a constraint: when predicting a word, the model is prevented from accessing future tokens. This is crucial for autoregressive models such as GPT, which generate text one token at a time.

Consider the sequence "abcde". In a naive training setup, GPT would be trained four times (on "a", "ab", "abc", and "abcd") to predict the next token. However, using masking, the entire sequence can be processed in parallel while still preserving the causal structure. The mask ensures that the model only attends to tokens up to and including the current one, effectively preventing it from "seeing" future tokens during prediction.

For instance, when predicting the token after "ab", the model sees only "a" and "b", not "c", "d", or "e". This differentiates GPT's masked self-attention from the bidirectional self-attention used in models like BERT [49]. Figure 13 visually illustrates this distinction.

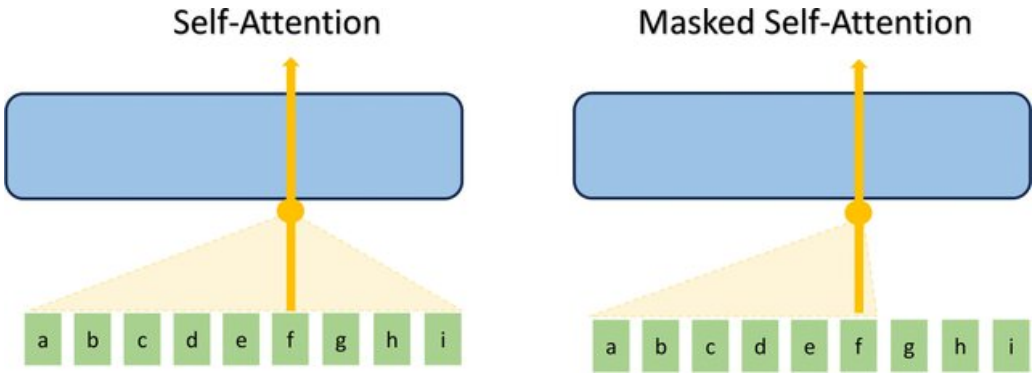

Figure 13: Comparison between Self-Attention and Masked Self-Attention

Multi-head self-attention further enhances this mechanism by allowing the model to attend to information from multiple representation subspaces simultaneously. In GPT-3, the self-attention mechanism is divided into $h = 96$ independent heads. Each head performs the self-attention steps independently with its own set of projection matrices. Each head works with a reduced dimension $d = d_{model}/h = 12,288/96 = 128$.

Each head produces an output matrix $Y_s \in \mathbb{R}^{2048\times128}$, and the outputs of all heads are concatenated to form the final self-attention output:

$$Y = \text{Concat}(Y_1, Y_2, ..., Y_{96}) \in \mathbb{R}^{2048\times12,288}$$

This multi-headed mechanism allows the model to simultaneously capture different types of relationships between words in the input sequence, significantly enhancing its expressiveness and contextual understanding.

Feedforward Neural Net

In addition to the masked multi-head self-attention mechanism, each transformer block in the GPT architecture contains a fully connected *feedforward neural network*. This component applies a non-linear transformation to each token representation independently, enabling the model to capture more complex patterns and feature interactions that go beyond the capabilities of attention alone.

The feedforward network typically consists of two linear transformations with a non-linear activation function between. Although it operates independently at each position in the sequence, it significantly increases the expressive power of the model and contributes to deeper learning of the representation.

Add & Norm

To stabilize training and improve generalization, each sublayer within the transformer block (both self-attention and feedforward) is followed by an Add & Norm operation. This consists of two parts:

- **Add:** A residual connection (or skip) adds the input of the sublayer to its output. This helps address the vanishing gradient problem, which is common in deep networks, and enables stacking many transformer layers without degradation in performance.
- **Norm:** Layer normalization is applied to the summed output. Unlike batch normalization, which relies on statistics computed across mini-batches, layer normalization is applied independently to each sample. Standardizes the values using the mean and variance of the individual input, making it robust to varying batch sizes and improving training stability.

Together, these operations help maintain the flow of gradients, prevent overfitting, and contribute to the architectural scalability of transformers.

3.3.4 Softmax

After the input passes through 12 transformer layers, the resulting matrix $Y \in \mathbb{R}^{2048\times12,288}$ contains rich, context-aware representations of each token in the sequence. These representations are then used to predict the next token through a final projection and softmax operation.

The model multiplies the output Y by a learned weight matrix $W \in \mathbb{R}^{12,288\times50,257}$, where 50,257 corresponds to the size of the GPT-3 vocabulary. This produces a logit matrix of shape $\mathbb{R}^{2048\times50,257}$, essentially assigning a raw score to each token in the vocabulary for each position in the input.

To convert these scores into probabilities, the softmax function is applied:

$$\text{Softmax}(YW)$$

This operation normalizes the logits for each token position so that they sum to 1, producing a probability distribution over the vocabulary. The token with the highest probability is then selected as the next predicted word in the model. This mechanism is repeated autoregressively during generation, with each newly predicted token fed back into the model to produce the next one.

3.4 Types of LLMs

To help in the selection of the most appropriate model for a specific need, LLMs can be categorized based on their underlying architectural design. In general, LLMs fall into three main types: *encoder-only*, *encoder-decoder*, and *decoder-only* models.

This classification is based on the transformer architecture, which serves as the foundational framework for modern LLMs [19]. The architecture is composed of two main components: the encoder, which processes and extracts meaning from input sequences, and the decoder, which generates output sequences based on encoded representations [102]. Depending on the task requirements, models may utilize either or both of these components.

In encoder-decoder models, both modules are integrated to support a wide range of applications, including translation and summarization [103]. Alternatively, encoder-only models are typically optimized for understanding and classification tasks, while decoder-only models are designed for text generation.

3.4.1 Encoder-only

Encoder-only models are designed primarily for tasks that require understanding and interpretation of input text. These models excel in NLU applications, such as sentiment analysis, named entity recognition, question answering, and text classification. Rather than generating new text, they focus on extracting semantic meaning from the input.

Due to their architecture, encoder-only models are particularly efficient at taking advantage of bidirectional context, which, as explained in the previous section, they take into account both preceding and following words when encoding a given token. This bidirectionality contributes to their strong performance in comprehension tasks. However, these models are generally not well suited for generative tasks and may face computational limitations when scaling to long input sequences.

Prominent examples of encoder-only models include BERT [49], DeBERTa [104], ELECTRA [105], ALBERT [106], RoBERTa [107], and XLM-RoBERTa [108], which have all demonstrated state-of-the-art performance in a wide range of NLU benchmarks.

3.4.2 Encoder-Decoder

Encoder-decoder LLMs integrate both encoder and decoder components, combining the strengths of each to form a highly versatile architecture. This dual structure enables the model to effectively handle tasks that require both a deep understanding of input text and coherent generation of output. As a result, encoder-decoder models are particularly well suited for applications such as machine translation, text summarization, and question answering.

The encoder processes the input sequence to generate a rich and contextual representation, which is then used by the decoder to produce the corresponding output sequence. This sequential flow supports a wide range of language tasks by aligning comprehension and generation within a unified framework.

However, this flexibility comes at a price. The complexity of coordinating both the encoding and decoding stages can make the training process more computationally demanding. Inference may also be slower compared to encoder-only or decoder-only models, especially when dealing with long sequences or real-time applications.

Notable examples of encoder-decoder models include T5 [109], which frames all NLP tasks as text-to-text problems; T0 [110], a multitask variant trained with instructions; LLaMA [92], a family of foundational models; PaLM [91], known for its massive scale and few-shot capabilities; Anthropic's Constitutional AI models [111]; and AlphaCode [112], designed for competitive programming and code generation.

3.4.3 Decoder-only

Decoder-only LLMs are specifically designed for text generation, making them particularly effective for tasks such as content creation, conversational agents, and creative text generation. These models excel in producing coherent and contextually appropriate language, especially in few-shot learning scenarios where minimal prompting is used to guide the output.

The decoder-only architecture processes input tokens sequentially and generates output one token at a time, conditioning each new prediction on the tokens generated so far. This autoregressive nature enables high-quality text generation, but also introduces certain trade-offs. In particular, decoder-only models can be less efficient in understanding the input context more deeply, particularly when the input is long or complex. Additionally, because of their size and sequential generation process, these models often require substantial computational resources during inference.

Prominent examples of decoder-only models include Codex [113], which specializes in code generation; GPT-3 [25], a foundational model known for its few-shot and zero-shot capabilities; LLaMA 2 [92] and LLaMA 3 [114], developed as efficient open-weight alternatives for large-scale language modeling; and Gemma [115], a family of lightweight open models optimized for general-purpose text generation.

3.5 Benchmarks

Quantitative AI benchmarks play a central role in the development, evaluation, and marketing of newly released AI systems. Alongside qualitative evaluation methods, benchmarks are widely regarded as providing essential feedback signals on the performance and capabilities of LLMs [116]. Their importance has grown to the point where companies invest substantial resources to achieve strong benchmark results. For instance, market leaders such as OpenAI are estimated to have spent hundreds of thousands of dollars in compute resources to secure high scores on the ARC-AGI benchmark [117].

In this work, we adopt the definition of benchmarks proposed by Deborah Raji et al.

[118], according to which a benchmark is understood as a combination of test datasets and performance metrics that are taken to represent general or specific tasks. Benchmarks are thus used as reference points to compare AI model capabilities and/or risks in a standardized and reproducible manner.

3.5.1 Cautions

Although benchmarks enable comparative evaluation of multiple AI models, several important cautions must be taken into account. The following issues are not listed in order of importance or urgency. Moreover, they should not be viewed as isolated problems, but rather as deeply interconnected challenges.

- **Data Collection, Annotation, and Documentation:** This relates to a broader critique of the lack of sufficient documentation in AI research, which lies at the core of ongoing calls for more transparent and trustworthy algorithmic systems [119]. Studies have shown that it is often difficult to accurately determine how, when, and by whom benchmark datasets were created [120]. This opacity undermines the reliability and generalization of benchmarks, making it harder to ensure that they can be applied in robust and meaningful ways [121].
- **Construct Validity and Epistemological Claims:** As pointed out by Raji et al. [118], many benchmarks suffer from construct validity issues, in the sense that they do not always measure what they claim to measure. This is particularly problematic when benchmarks purport to assess universal or general capabilities, since this can vastly misrepresent their actual scope and reliability.
- **Sociocultural Context and Gaps:** Qualitative research has examined the cultural and social environments in which benchmarks are created, showing that they are often shaped by shared but arbitrary assumptions, commitments, and dependencies [122]. This raises questions about their universality and applicability across different contexts.
- **Narrow Benchmark Diversity and Scope:** Previous research has shown that current benchmarking practices suffer from diversity issues, a problem also prevalent within the broader AI ecosystem [123]. Moreover, the vast majority of benchmarks are focused on text-based tasks, while other modalities such as audio, images, video, and multimodal systems remain largely underexplored [124].
- **Economic, Competitive, and Commercial Roots:** Benchmarks are not neutral evaluation tools but are often deeply embedded in competitive and commercial dynamics. Prior research has highlighted how capability-oriented benchmarks are closely tied to corporate marketing strategies, playing an important role in amplifying AI hype, attracting customers and investors, and showcasing performance advantages over competitors [125].

- **Rigging, Gaming, and Measures Becoming Targets:** Researchers have noted strong incentives to rig benchmark tests, especially in domains lacking established evaluation practices. Techniques for scoring highly are widely circulated online, and models are often optimized for benchmark-specific formats, such as multiple-choice questions, sometimes even simulating ethical or safety alignment [126].

- **Dubious Community Vetting and Path Dependencies:** Benchmarks often become de facto standards not through rigorous vetting but through citation culture and academic momentum [127]. For example, ImageNet and the Lena test image gained prominence through historical accidents rather than principled evaluation. Once widely cited, benchmarks reinforce their own authority, a phenomenon sometimes described as "peer-washing" [128].

- **Rapid AI Development and Benchmark Saturation:** The rapid pace of AI development has rendered many benchmarks outdated, as they were designed for models far less capable than today's systems [129]. Benchmarks are often quickly saturated, with models achieving near-perfect scores, undermining their discriminative power. Evaluation frameworks can also be slow and resource-intensive, leading to delays in assessing new models, which is especially problematic in regulatory contexts where timely assessments are crucial [130].

- **AI Complexity and Unknown Unknowns:** A final limitation concerns the intrinsic complexity of AI systems and the unpredictability of their risks. Benchmarks are constrained by the knowledge of their creators and may fail to capture emerging capabilities that surpass human foresight [131]. Latent vulnerabilities may remain undetected, raising concerns that models appearing safe might in fact be unsafe. For instance, simple prompts have been found to bypass safety mechanisms and even leak training data [132].

3.5.2 Graduate-level Google-Proof Q&A

As presented in Section 1.3, one of the most prominent and widely discussed benchmarks is the *Graduate-level Google-Proof Q&A* (GPQA), introduced by Rein et al. [10]. GPQA is designed to evaluate reasoning and problem-solving abilities in advanced, domain-specific contexts that cannot simply be solved by retrieving answers from the internet (hence the term *Google-proof*). The benchmark consists of graduate-level multiple-choice questions spanning diverse disciplines such as physics, chemistry, and biology, with a focus on testing deep conceptual understanding rather than surface-level memorization.

Unlike conventional knowledge-retrieval benchmarks, GPQA explicitly measures an AI model's ability to engage in multi-step reasoning, to synthesize information across subfields, and to correctly rule out distractor answers that may appear plausible at first glance. This makes GPQA particularly relevant for evaluating large language models that claim to perform complex reasoning tasks. However, the benchmark also highlights current limitations: even the most advanced frontier models struggle to approach human-level performance, often scoring closer to chance than to expert accuracy.

3.5.3 Frontier Math

Another well-established benchmark for evaluating reasoning in mathematics and formal sciences is *Frontier Math*. This benchmark contains challenging, competition-level mathematical problems that require symbolic manipulation, abstract reasoning, and the ability to follow logical chains of deduction. Frontier Math has been used to assess how well LLMs generalize beyond natural language into precise, highly-structured problem domains where small mistakes can lead to incorrect solutions.

Figure 14 illustrates the performance of several state-of-the-art models on Frontier Math. As can be observed, current models do not achieve high scores, which emphasizes the difficulty of the benchmark. The strongest performance is obtained by GPT-5 (high capacity), with an accuracy approaching 25%. This is followed by models such as o4-mini (medium) and GPT-5 (medium), which perform somewhat lower. Competing models from other organizations demonstrate more modest results: for instance, Grok 4 achieves a score around 12%, while Gemini 2.5 Pro performs at approximately 11%.

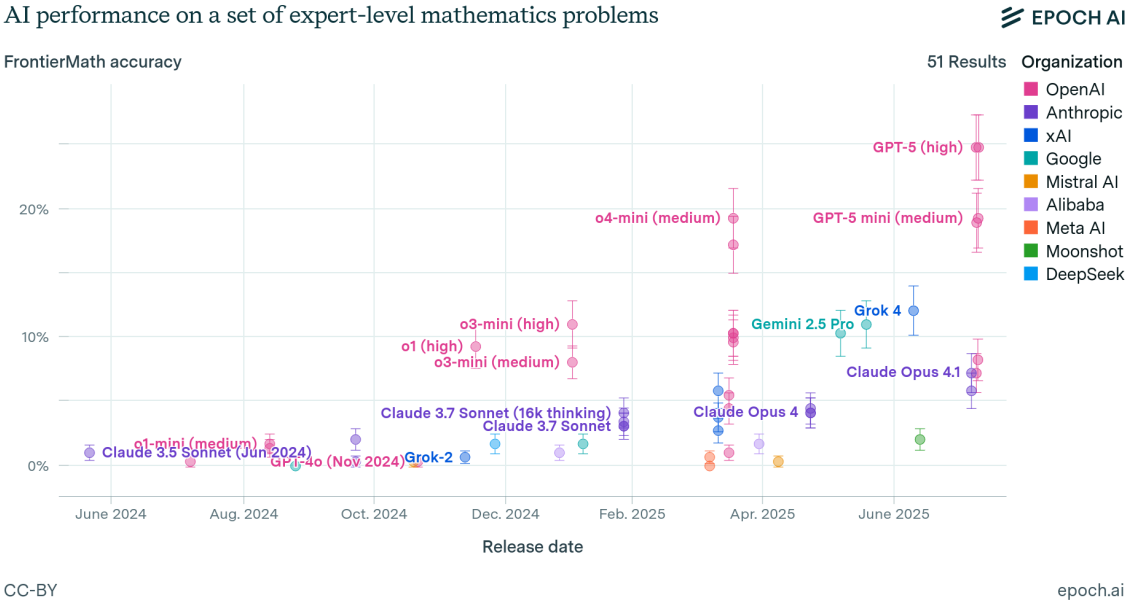

Figure 14: Performance of AI systems on Frontier Math in recent years

These results reveal two important insights. First, despite their rapid progress, even the most advanced models still underperform on tasks requiring rigorous mathematical reasoning, highlighting a major bottleneck for general-purpose AI. Second, the significant performance gap between different providers underscores that progress in mathematics-focused benchmarks is not yet uniform across the AI ecosystem. In practice, this suggests that AI remains far from being a reliable assistant in domains where exactness and logical consistency are non-negotiable, such as higher mathematics, engineering, or formal verification.

3.6 Applications

LLMs have proven highly effective across a wide range of domains. This section highlights notable applications in various fields to illustrate their growing impact and versatility.

3.6.1 Software Engineering

LLMs have shown impressive capabilities in numerous software engineering tasks, including defect prediction [133], code review [134], code generation [135], and the analysis of software-related queries [136]. These applications have contributed to significant improvements in code quality and debugging efficiency, while also reducing the level of manual intervention required during software development.

One of the most influential models in this area is Codex [113], which is built on a large generative pre-trained model comprising up to 12 billion parameters. Codex assists developers by automatically generating code, suggesting optimizations, and identifying as well as correcting errors. Its introduction marked a turning point in the integration of natural language processing models into software engineering workflows.

Building upon the foundation laid by Codex, a number of specialized models have been introduced to address specific challenges within programming and software development. Notable contributions include DeepMind's AlphaCode [112], designed to solve problems in programming competitions, Meta's InCoder [137] and Code Llama [138]. Additional efforts such as Salesforce's CodeRL [139] and CodeGen [140] further demonstrate the diversity and scope of models tailored to various programming contexts.

These advances are rapidly transforming the landscape of automated code generation, establishing new benchmarks, and enabling higher levels of productivity and automation in software engineering. A particularly striking example of this shift is that more than 25% of Google's code is now written with the assistance of AI technologies, a figure that, according to Google's CEO Sundar Pichai, represents only the beginning of a larger transformation [141].

3.6.2 Drug Discovery

Drug discovery is a field traditionally associated with considerable complexity, long development cycles, and high financial costs. The adoption of computational methodologies such as AI, DL, and quantum mechanical approaches, has opened new possibilities to accelerate the identification of potential drug candidates and predict their pharmacological properties [142].

Within this evolving landscape, LLMs, such as ChatGPT, have emerged as valuable tools due to their ability to process, generate, and analyze natural language text [143]. Their integration into drug discovery workflows has introduced novel opportunities to

improve efficiency, particularly in the early stages of research.

In the initial phases of drug discovery, LLMs contribute by helping to identify suitable drug targets for various diseases [144]. Through their ability to extract and synthesize relevant biomedical knowledge from a wide range of text corpora, these models provide preliminary insights into the structure and function of protein-based targets. Although these findings require further validation through experimental or alternative computational methods, LLMs offer a useful foundation for narrowing down potential therapeutic targets.

Beyond the target identification stage, LLMs also play an important role in the subsequent steps of drug development. They support the design and virtual screening of small molecules that may interact effectively with identified targets [143]. By facilitating the exploration of chemical space and aiding in molecular generation and filtering, LLMs help streamline the discovery process, ultimately contributing to more efficient and data-driven pharmaceutical research.

3.6.3 Finance

LLMs are driving substantial advancements across the financial sector. Key applications include algorithmic trading and portfolio management [145], financial risk modeling [146], financial text mining [147], as well as financial advisory services and customer support [148].

The integration of artificial intelligence into financial advisory and customer service is a rapidly evolving area. As reported by Chiara Valentina et al. [149], AI-powered chatbots already perform over 37% of support functions in various e-service contexts. In the financial domain, these systems are increasingly being adopted as efficient and cost-effective alternatives to traditional human-operated customer service channels, a trend highlighted in the "Chatbots in Consumer Finance" report [150].

The emergence and continued improvement of LLMs have further extended the range of complex tasks that AI systems are capable of handling within the financial industry. From interpreting market sentiment in real time to automating compliance and generating analytical reports, the applications of LLMs are steadily expanding. As noted in recent studies [151], this progression underscores the transformative potential of LLMs in enhancing both operational efficiency and decision-making across various financial services.

3.6.4 Medical

In the medical domain, artificial intelligence is playing an increasingly important role in improving clinical workflows, supporting decision-making, and accelerating research and education [152]. LLMs, in particular, have shown strong potential in understanding

complex medical texts, answering clinical questions, and reasoning over biomedical data.

A widely used benchmark for evaluating language models in medicine is the MedQA dataset [153], which contains multiple-choice questions modeled after the United States Medical Licensing Examination (USMLE). These questions test a model's ability to understand clinical scenarios and apply medical knowledge.

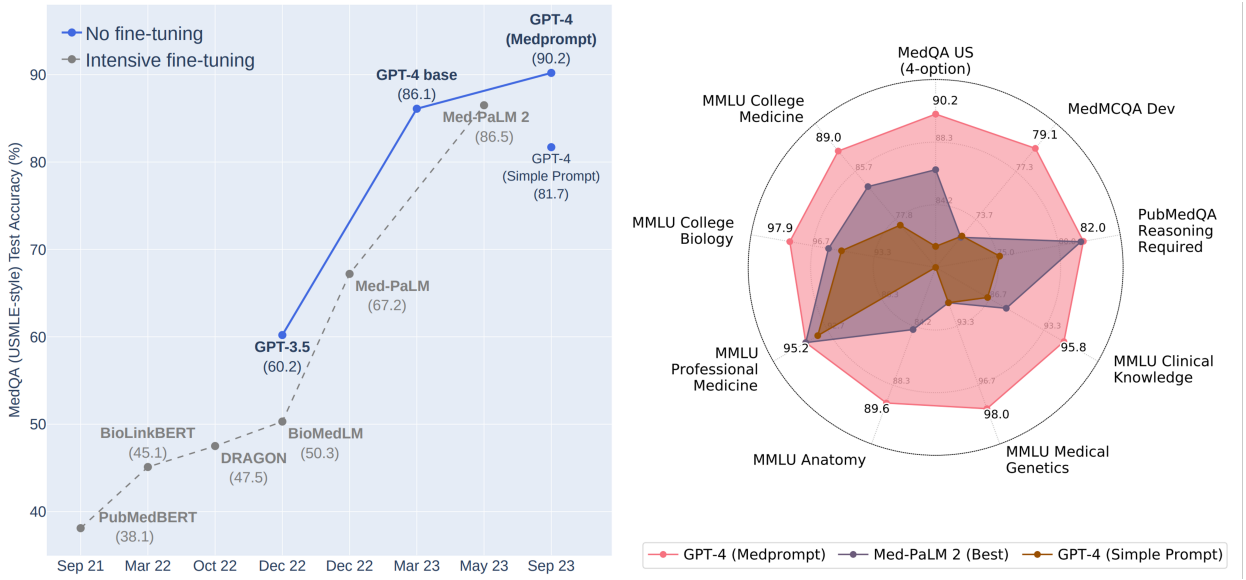

Figure 15: Performance of LLMs on biomedical benchmarks

As shown in Figure 15, GPT-4 achieves a remarkable score of 90.2% on MedQA when used with a specialized prompting strategy called Medprompt. This performance surpasses both earlier versions, such as GPT-3.5, which achieved 60.2%, and fine-tuned models such as Med-PaLM 2, which reached 86.5%. The graph on the left of the figure illustrates the progression of different models over time, differentiating those with intensive domain-specific fine-tuning (gray) from general-purpose models that rely on prompting (blue). In particular, GPT-4 without any fine-tuning already achieves competitive results, highlighting the strength of prompt-based strategies.

The right-hand side of the figure compares performance across several biomedical and clinical benchmarks, including MedMCQA, PubMedQA, and multiple MMLU subdomains such as clinical knowledge and anatomy. GPT-4 with Medprompt consistently outperforms both Med-PaLM 2 and GPT-4 using a simple prompt on all tasks. These results demonstrate not only the capabilities of general-purpose models such as GPT-4 in specialized domains but also the importance of how prompts are structured. As highlighted in a recent Microsoft study [154], prompt engineering is emerging as a key technique for unlocking the full potential of large models, an idea explored in more depth later in this work.

3.6.5 Legal

Within the legal domain, LLMs have shown considerable promise across a range of tasks, particularly in legal text generation, document drafting, and case prediction. For example, ChatGPT has demonstrated the ability to reason through legal scenarios by achieving a 50.3% accuracy rate on NCBE Multistate Bar Examination (MBE) practice test and obtained passing scores in legal subjects such as Evidence and Torts [155]. These results highlight the potential of LLMs to engage meaningfully with legal content, even in highly structured and rule-bound contexts.

The legal industry has begun experimenting with LLMs in practical settings. Notably, an Australian law firm utilized ChatGPT to draft a legal statement based on the 1992 Mabo case. The resulting document was comparable to what might be expected from a first-year law associate, suggesting a future in which AI could assist with routine drafting and research tasks [156]. Nevertheless, while such models offer productivity gains, they also raise concerns around accuracy and reliability. Iu et al. [157] conducted a study to assess whether ChatGPT could fully replace legal professionals. Their findings concluded that while ChatGPT holds value as a supporting tool, it is not suited to operate independently due to its inability to meet the high standards of precision, reliability, and up-to-date knowledge required in legal practice.

One of the key limitations observed in LLMs, particularly in sensitive domains like law, is the phenomenon known as hallucination. Hallucinations occur when a language model generates plausible-sounding but factually incorrect or entirely fabricated information. This is especially problematic in legal settings where factual accuracy and reference to current legislation or precedent are critical. To address this issue, newer systems such as DeliLaw [158] have incorporated retrieval-augmented generation techniques that connect the model to external legal databases, significantly reducing hallucination rates and improving factual grounding. In addition, purpose-built legal LLMs are being developed to better align with the specific demands of the legal profession. Recent examples include ChatLaw [159], which adopts a multi-agent, collaborative framework to improve legal reasoning and document drafting.

These advancements indicate that while LLMs are unlikely to fully replace legal experts, their integration into legal workflows, as research assistants, drafting aids, or initial reviewers, could reshape the profession by augmenting human capabilities and improving efficiency.

3.6.6 Education

The rapid advancement of artificial intelligence is significantly reshaping the landscape of education, offering new ways to enhance both teaching and learning. Among these developments, LLMs have attracted considerable attention for their potential to transform educational environments. Their versatility allows them to support various instructional strategies and help address long-standing challenges in the field.

LLMs can be used to simulate student behavior, a valuable tool for teacher training and classroom preparation [160]. By mimicking typical classroom interactions, these models provide future educators with realistic scenarios to practice teaching strategies and classroom management. In addition, LLMs can serve as virtual instructors, providing personalized instruction that adapts to the learners pace, needs, and level of knowledge of individual students [161].

Their role as personal tutors is particularly promising. LLMs can offer immediate feedback on assignments, provide customized evaluations, and suggest appropriate learning resources based on a student's progress [162, 163]. This level of personalization supports more effective learning pathways and encourages learners to take a more active role in their education.

Beyond individual instruction, LLMs also help improve student engagement and autonomy. Their interactive nature can promote curiosity, critical thinking, and deeper interest in the subject matter [164]. What is more, they offer a scalable solution to broader systemic issues in education, such as high student-to-teacher ratios. By assisting human instructors and providing additional support, LLMs can help bridge the gaps in access to quality education [165].

Integrating LLMs into educational settings represents a unique opportunity to scale, personalize, and democratize learning. As these models continue to evolve, they have the potential to make high-quality educational experiences more accessible, adaptive, and inclusive for diverse learner populations around the world.

3.7 Drawbacks and Future Directions

Although LLMs offer substantial benefits across professional and personal domains, it is equally important to consider the limitations and challenges they present. This section outlines some of the key drawbacks associated with current LLM technology, many of which also point to future directions for research and development.

3.7.1 Privacy

Despite their widespread adoption in multiple sectors, LLMs raise significant privacy concerns [166]. One of the most pressing issues is the potential for data leakage. Due to the vast and often uncurated nature of training datasets, these models can inadvertently memorize and later reproduce sensitive or personally identifiable information. This poses serious risks, especially in highly sensitive fields such as healthcare or finance, where breaches of confidentiality could have legal and ethical consequences [167].

In addition, many datasets used to train LLMs are not fully anonymized. Personal identifiers or confidential information can become embedded within the model, increasing the probability of unintentional disclosure [168]. Such risks bring LLM development

into tension with data protection laws, such as the General Data Protection Regulation (GDPR) [169], which mandates strict handling of personal data and provides individuals with rights over how their information is used.

To address these challenges, recent research has proposed solutions, such as machine learning unlearning frameworks. For example, the work of Domingo-Ferrer et al. [170] introduces the Efficient Unlearning with Privacy Guarantees (EUPG) framework. This approach aims to formally guarantee that specific data used in training can be effectively removed from the model, thereby enhancing user privacy.

In addition to unintentional risks, the misuse of LLMs for malicious purposes presents a growing concern. Emerging tools such as FraudGPT and WormGPT [171] exemplify this threat. These models are explicitly designed for cybercrime, capable of generating phishing emails, suggesting malicious link placements, and assisting in advanced cyber-attacks such as Business Email Compromise. Trained in data specifically focused on malware and fraud, they demonstrate how the same underlying technology can be weaponized when placed in the wrong hands.

These developments highlight the urgent need for robust privacy preservation techniques, responsible AI governance, and clear regulatory frameworks to prevent misuse and ensure that LLMs are deployed safely and ethically. As LLM adoption continues to grow, addressing privacy risks will remain a central challenge for the field.

3.7.2 Fairness

LLMs are increasingly used in decision-making processes across various domains, ranging from hiring to education and public services. However, their outputs can reflect and even amplify societal biases, disproportionately affecting marginalized or underrepresented groups. For example, studies have shown that when LLMs are used to screen résumés for programming positions, they can favor male candidates over female candidates, revealing a clear pattern of gender discrimination [172].

This kind of bias typically originates from the data on which these models are trained. LLMs are often trained on vast, unfiltered collections of text gathered from the Internet. As a result, they absorb and replicate harmful stereotypes, misrepresentations, exclusionary behaviors, and even derogatory language embedded in those datasets [173]. These biases are not evenly distributed, they tend to reinforce existing structural inequalities, and disproportionately affect individuals from already disadvantaged communities.

Efforts to mitigate such biases have typically focused on preprocessing strategies, such as removing biased examples from training datasets. Although this may reduce certain explicit forms of bias, it is often insufficient and can even compromise the overall performance of the model in language understanding and generation tasks [174]. The bias in LLMs is complex and deeply embedded, which means that simple data-cleaning techniques alone are unlikely to completely resolve the problem.

As Saxena et al. [175] argue, these challenges are rooted not only in the technical design of LLMs but also in broader social contexts and contested definitions of fairness. Addressing bias requires more than a technical solution-oriented approach to design. Researchers and developers must consider the needs and perspectives of marginalized communities from the very beginning of the design process, rather than reacting to bias only after harm has been observed [176]. Proactive, inclusive and interdisciplinary interventions are essential to ensure that LLMs promote fairness and benefit all users equitably.

3.7.3 Safety

As artificial intelligence becomes increasingly integrated into industrial and professional applications, concerns regarding safety and reliability have become more pronounced. While LLMs offer powerful capabilities, they also introduce notable security and trustworthiness challenges. One of the most pressing issues is the generation of outputs that deviate from user intent, context, or factual accuracy, a phenomenon commonly referred to as "hallucinations," as previously discussed in the subsection on legal applications. Such errors significantly undermine the reliability of LLMs, particularly in high-stakes fields such as medicine, where misinformation can lead to serious consequences [177].

To address these risks, a widely adopted approach involves incorporating humans into the training loop through Reinforcement Learning from Human Feedback (RLHF) [178]. This method aims to better align the model's behavior with human values and expectations by optimizing responses based on human-annotated preferences. In practice, RLHF has been shown to improve model safety, especially when the feedback process includes explicit safety-related signals, as demonstrated in the development of GPT-4 [90].

However, the effectiveness of RLHF is highly dependent on the availability of high-quality human feedback, typically provided by professional annotators. This dependency introduces substantial costs and logistical constraints. Consequently, there is a growing need to enhance RLHF frameworks to reduce the burden on human annotators while maintaining feedback quality. One promising direction involves leveraging LLMs themselves to assist in the annotation process. When combined with robust oversight, these models can streamline feedback generation, reduce the reliance on manual annotation, and contribute to improved safety outcomes.

As LLMs deployments expand into critical domains, ensuring safety through well-aligned training strategies and scalable human-in-the-loop approaches will be fundamental to their responsible and effective use.

3.7.4 Intellectual Property

LLMs possess the ability to generate content that closely resembles human-created material, raising serious concerns regarding intellectual property (IP) rights. These models are trained in vast corpora that often include copyright-protected, proprietary, or oth-

erwise sensitive textual data. Such training data may inadvertently encompass personal information, protected knowledge, patented technologies, and other forms of intellectual property, making it difficult to ensure that the output respects the rights of the original creators.

A prominent example of this issue is found in AI-generated code. Code generation models trained in public open-source repositories can reproduce code snippets that strongly resemble, or even exactly match, existing code, often without preserving licensing terms [179]. This has led to legal action against Microsoft, GitHub, and OpenAI, with plaintiffs alleging that GitHub Copilot reproduced licensed code without proper attribution or compliance with licensing conditions [180].

Beyond software, similar challenges exist in the realm of LLM-generated text and images. Recent lawsuits filed in July 2023 by a group of well-known novelists claimed that their books had been used without authorization to train LLMs developed by OpenAI and Meta [181]. Another high-profile case emerged in December 2023 when The New York Times filed a lawsuit against OpenAI and Microsoft, accusing them of using copyrighted news articles to train their generative models without proper consent or compensation [182].

One of the most recent controversies in this space involves the rise of AI-generated music. In July 2025, a band called *Velvet Sundown*, composed entirely of music generated by artificial intelligence, achieved more than a million streams on Spotify [183]. Although the creators of the band were transparent about its synthetic nature, the incident sparked a widespread debate among music industry professionals. Many expressed concern that listeners were not sufficiently informed that the music was AI-generated, calling for clearer labeling standards to avoid misleading audiences.

A more recent development in AI-generated creative content involves the ability of tools such as DALL-E·3, integrated into ChatGPT, to generate images that convincingly mimic the visual styles of revered artists and studios. In early 2025, a viral trend emerged around the creation of Studio Ghibli–style illustrations, prompting millions of users to generate Ghibli-esque visuals, sparking massive engagement on platforms such as social media and even prompting servers to struggle under the load [184]. Although OpenAI claims to block direct requests to reproduce the styles of living artists, broader "studio-style" prompts remain available, resulting in images that closely emulate famous aesthetics without artist attribution.

In the current digital landscape, there is an urgent need for more robust attention and regulatory measures concerning intellectual property rights in the context of LLMs and generative AI. As illustrated in Figure 16, users now have the ability to create highly realistic and stylistically nuanced visual content using tools like ChatGPT and DALL·E 3. Although this technological advancement opens new avenues for creative expression, it simultaneously introduces challenges related to the unauthorized reproduction or emulation of artistic styles, designs, and copyrighted works. To address these concerns, it is essential to establish regulatory frameworks that both protect the legitimate rights and interests of original creators and users, and support responsible innovation in the

development and deployment of LLM technologies [185].

Figure 16: Images generated by users of ChatGPT and DALL-E 3

4 Development of the CSXSC Model

Building upon the theoretical foundations of the previous chapter, this section details the practical development of a sentiment classification model specifically tailored for Catalan-language social media content. This model, designated as the **Classificador de Sentiments a Xarxes Socials en Català (CSXSC)**, is designed to classify text into one of three sentiment categories: *positive, negative, or neutral*.

The methodology followed a structured, multi-stage approach. The process began with the construction of a high-quality, manually curated dataset to capture the unique linguistic nuances of Catalan on social media. This was followed by the selection of a suitable pre-trained model for fine-tuning, the execution of the training process, and finally, a comprehensive evaluation of the model's performance using a suite of established metrics.

To ensure transparency and facilitate the replication of this study, all code, data processing scripts, and experimental configurations have been made publicly available. The complete workflow is documented across three Jupyter Notebooks accessible via a GitHub repository[3]. Furthermore, the final trained CSXSC model has been uploaded to the Hugging Face Hub, making it an open-access resource for future research and application[4]. All experiments were conducted on a laptop equipped with an **NVIDIA RTX 3060** consumer-grade GPU.

4.1 The Aina Project and Aina Kit

To construct the dataset and select an appropriate model for fine-tuning, this work leverages the resources provided by the Aina Project[5]. The Aina Project is a major initiative that provides open-access linguistic resources, corpora, and computational models to facilitate the development of AI-based applications in Catalan. This initiative is led by the Government of Catalonia in collaboration with the Barcelona Supercomputing Center (BSC-CNS) and forms part of the Artificial Intelligence Strategy of Catalonia (Catalonia.AI).

The practical implementation of this initiative is the Aina Kit[6], a comprehensive and organized collection of the open-source models and datasets developed by the project. It serves as a toolkit for individuals and organizations aiming to build AI-powered products and services in the Catalan language.

The Aina Kit is composed of several key components:

[3] https://github.com/Danie1Arias/CSXSC/tree/main
[4] https://huggingface.co/Danie1Arias/csxsc
[5] https://projecteaina.cat/
[6] https://langtech-bsc.gitbook.io/aina-kit

- **Language Models:** A suite of foundational and fine-tuned models for various NLP tasks, including text generation and classification, speech recognition and synthesis, and machine translation.
- **Datasets:** A diverse collection of general and specialized corpora (text, speech, and translation) that were used for training and evaluating the language models.
- **Integration Tools:** Software and utilities designed to facilitate the integration of the Aina Kit's resources into new or existing applications and workflows.
- **Documentation and Demonstrators:** Comprehensive adoption guides, FAQs, and practical examples (demonstrators) that showcase the application of the models and assist with their implementation.

The remainder of this chapter will provide a detailed exposition of the strategies, datasets, and models selected for this study. A thorough justification for the use of specific Aina Kit resources will be presented in the relevant sections.

4.2 Dataset

The foundation of an effective sentiment analysis model is a high-quality, representative dataset. For the context of Catalan social media, it is crucial that this dataset captures the nuances of real-world communication, including colloquialisms, dialectal variations, and the specific contextual cues characteristic of online platforms. To meet this requirement, we constructed a custom dataset by integrating and adapting materials from trusted, third-party sources. Each source dataset was subjected to specific preprocessing strategies to ensure the final labels for sentiment were both consistent and meaningful.

Our final, consolidated dataset is constructed from three distinct sources. The first two components are derived from the Aina Project, while the third is an adaptation of the GoEmotions dataset [186]. Developed by Google AI, GoEmotions is a large-scale, human-annotated corpus of 58,000 English Reddit comments. These are labeled with 27 distinct emotion categories, in addition to a neutral class. As each comment can be associated with one or more emotion labels, the dataset is inherently suitable for multilabel classification tasks.

This section outlines the characteristics of each source dataset and details the specific preprocessing strategies that were applied. The full implementation, including all data processing scripts and source code, is available in the project's public GitHub repository as a Jupyter Notebook[7] and the dataset publicly available on Hugging Face[8].

[7] https://github.com/Danie1Arias/CSXSC/blob/main/dataset.ipynb
[8] https://huggingface.co/datasets/Danie1Arias/sentiment-analysis-catalan-reviews

4.2.1 The GuiaCat Dataset

The first dataset used is **GuiaCat**[9], a corpus comprising 5,750 restaurant reviews written in Catalan. Each review is annotated with four numerical scores, their average, and a qualitative sentiment label. The data was originally sourced from GuiaCat and subsequently curated by the Barcelona Supercomputing Center (BSC) as part of the Aina Project.

Each review within the dataset contains the following fields:

- **Service:** A numerical score from 0 to 10 evaluating the quality of the restaurant's service.
- **Food:** A score from 0 to 10 evaluating the quality of the food.
- **Price-to-Quality Ratio (price-quality):** A score from 0 to 10 assessing the value for money.
- **Environment:** A score from 0 to 10 evaluating the restaurant's atmosphere.
- **Avg:** The arithmetic mean of the four preceding scores.
- **Text:** The user-written review in natural language.
- **Label:** A categorical sentiment descriptor derived from the average score, with five possible values: *molt bo*, *bo*, *regular*, *dolent*, and *molt dolent*.

For the purposes of this study, only the `text` and `label` fields were extracted to construct a three-class sentiment classification task. The original five sentiment levels were mapped to our simplified schema as follows:

Numerical Rating	Original Label	Classification
> 8	Molt bo	Positive
$6 \leq$ rating ≤ 8	Bo	Positive
$4 \leq$ rating < 6	Regular	Neutral
$2 \leq$ rating < 4	Dolent	Negative
< 2	Molt dolent	Negative

Table 1: Mapping of numerical ratings to original labels and final sentiment categories.

The processed dataset, containing all 5,750 reviews, was compiled into a single CSV file (`guiacat.csv`). An analysis of the sentiment labels reveals the following distribution: 94.3% Positive, 3.6% Negative, and 2.1% Neutral. This distribution demonstrates a significant class imbalance, with the vast majority of reviews skewed towards positive sentiment.

[9] https://huggingface.co/datasets/projecte-aina/GuiaCat

4.2.2 The Catalan Structured Sentiment Analysis Dataset

The second corpus integrated into our study is the **Catalan Structured Sentiment Analysis (CaSSA)**[10] dataset. This collection comprises 6,400 reviews and forum posts annotated with fine-grained sentiment information. Specifically, each text is annotated with all constituent *polar expressions*, resulting in a total of 25,453 annotated instances.

For each polar expression, the dataset provides the following structured information:

- **Source:** The entity expressing the sentiment.
- **Target:** The entity toward which the sentiment is directed.
- **Expression:** The specific textual fragment that conveys the sentiment.
- **Polarity:** The sentiment category (positive, negative, or neutral).
- **Intensity:** The strength of the sentiment (standard or strong).

Each record in the dataset consists of a textual entry, which may contain zero or more polar expressions grouped within an `opinions` field. To adapt this fine-grained dataset for our three-class classification task, a majority voting strategy is employed. For each text, the total number of positive, negative, and neutral polar expressions is counted. The general sentiment label for the text is then assigned based on the most frequent polarity. Entries with no polar expressions or those with an equal number of competing sentiments are classified as neutral.

The resulting dataset, compiled into the file `cassa.csv`, contains 6,400 processed entries. The final class distribution is 64% Positive, 9.5% Negative, and 26.5% Neutral. Similarly to the GuiaCat corpus, this distribution shows a predominance of positive sentiment, although the class imbalance is notably less pronounced.

4.2.3 The GoEmotions Dataset

The third and final source is the **GoEmotions** dataset [186], a large-scale, human-annotated corpus comprising 58,000 Reddit comments from popular English-language subreddits. Developed by Google AI, its primary goal is to support more nuanced emotion and sentiment classification in user-generated content.

Each comment is labeled with one or more emotions from a taxonomy of 27 distinct categories, plus a neutral class. This taxonomy was designed for both psychological validity and practical applicability. Unlike classical models based on Ekman's six basic emotions (which include only one positive sentiment), the GoEmotions taxonomy offers a much broader affective range:

[10] https://huggingface.co/datasets/projecte-aina/CaSSA-catalan-structured-sentiment-analysis

- **Positive emotions (12)**: Admiration, amusement, approval, caring, desire, excitement, gratitude, joy, love, optimism, pride, and relief.
- **Negative emotions (11)**: Anger, annoyance, disappointment, disapproval, disgust, embarrassment, fear, grief, nervousness, remorse, and sadness.
- **Ambiguous emotions (4)**: Confusion, curiosity, realization, and surprise.
- A single **Neutral category**.

As illustrated in Figure 17, positive emotions are the most frequent within the dataset.

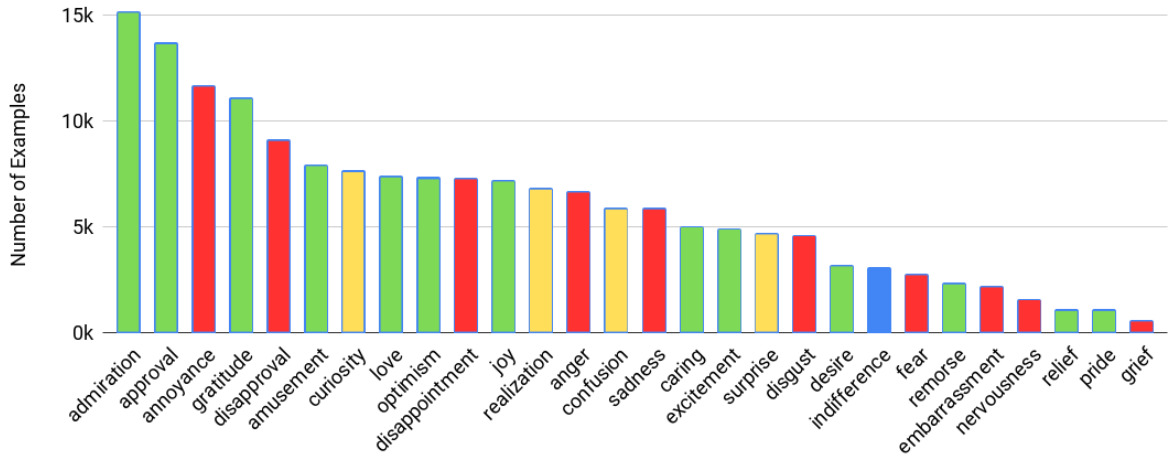

Figure 17: Number of examples per emotion in the GoEmotions dataset. In color green the positive emotions, in color red the negative, in yellow the ambiguous and, in color blue, the neutral emotion.

The combination of the GuiaCat and CaSSA datasets resulted in a preliminary corpus of 12,150 entries, which was heavily skewed towards positive sentiment (9,517 positive, 1,815 neutral, and 818 negative). To remedy this significant class imbalance, we strategically sampled instances from the GoEmotions dataset to augment our corpus and achieve a target distribution of approximately 40% positive, 30% negative, and 30% neutral.

To implement this, a sampling algorithm has been developed. For this study, the total number of positive reviews for the final dataset is set to 9,617 (the original 9,517 plus 100 new instances from GoEmotions). Using this figure as the 40% target, the required number of negative (30%) and neutral (30%) instances is calculated to be 6,394 and 5,397, respectively. These required instances were then sourced from the GoEmotions dataset. After determining the required number of instances for each sentiment class, the following two-step process is employed.

Step 1: Data Classification

To align the GoEmotions dataset with our three-class task, the 27 fine-grained emotion labels were mapped to broader sentiment categories. Each comment is assigned a single

sentiment label based on the predominant emotion type it contained. In cases of a tie between polarities, or if a comment contained only ambiguous emotions, it is classified as neutral.

Step 2: Data Translation

As GoEmotions comments are in English, a high-quality machine translation step is required. Although generic services like Google Translate are effective, they can fail to capture specific jargon or colloquialisms. Therefore, this study use the **Aina English-Catalan machine translation model** (aina-translator-en-ca)[11]. This model, provided by the Aina Kit, was trained from scratch using the Fairseq[12] toolkit on a filtered corpus of 30,023,034 English-Catalan sentence pairs.

As shown in Table 2, the Aina translator demonstrates competitive performance against other widely used services across various test sets.

Test Set	SoftCatalà	Google Translate	Aina Translator
Spanish Constitution	32.6	37.8	**41.2**
United Nations	39.0	40.5	**41.2**
AAPP	46.5	51.40	**51.70**
European Commission	49.1	**52.0**	51.0
Flores 200 dev	41.0	**45.1**	43.3
Flores 200 devtest	42.1	**46.0**	44.1
Cybersecurity	42.5	**48.1**	45.8
wmt19 biomedical	21.7	25.5	**26.7**
wmt13 news	34.9	**35.7**	34.0
Average	38.82	**42.45**	42.1

Table 2: Evaluation results on the machine translation from English to Catalan compared to Softcatalà and Google Translate

Step 3: Data Evaluation

Following the data classification and translation stages, an automated evaluation was implemented to assess the quality of the translated GoEmotions corpus. To automate this process, this study employed the **Salamandra-7B-Instruct** model[13], the 7-billion parameter, instruction-tuned variant of the Salamandra model family. This transformer-based, decoder-only language model was pre-trained from scratch on a highly curated corpus of text and code from 35 European languages.

The model was tasked with two sequential objectives for each translated text:

1. **Quality Assessment:** The model was instructed to rate the translation's quality

[11] https://huggingface.co/projecte-aina/aina-translator-en-ca
[12] https://fairseq.readthedocs.io/en/latest/
[13] https://huggingface.co/BSC-LT/salamandra-7b-instruct

on a scale from 1 (very poor) to 5 (excellent), while disregarding minor grammatical errors or informalities common in social media content.

> Prompt: *Ets un avaluador de qualitat de traducció en català. Et donaré un text en català procedent de xarxes socials, traduït automàticament des de l'anglès. Avalua la qualitat de la traducció en una escala del 1 (molt dolenta) al 5 (excel·lent). Ignora errors ortogràfics menors, abreviatures o estil informal. Tingues en compte que el text pot ser informal o col·loquial ja que prové d'opinions de xarxes socials. Respon només amb el número.*

2. **Justification and Filtering:** If a translation received a score below 3, the model was then prompted to provide a brief justification for the low rating. These entries were subsequently flagged for removal from the dataset.

> Prompt: *El text següent és una traducció automàtica del anglès al català. La seva qualitat s'ha valorat amb una puntuació baixa ($\leq$ 2) en una escala de 1 a 5. Explica breument per què podria ser de baixa qualitat, centrant-te en problemes de traducció i no en el contingut.*

Given the significant computational cost of inference with this model, a random sample of 500 translated reviews was processed for this automated evaluation, which required approximately 10 hours of computation time.

The automated process proved to be effective in identifying genuinely poor translations. Below are examples of entries correctly discarded by the model.

- Text: `Tose dm de cvs igual que delsym`
- Reason: *La traducció automàtica d'aquesta frase anglesa "Tose dm de cvs igual que delsym" ha estat considerada de mala qualitat perquè conté errors gramaticals i ortogràfics. Concretament, hi ha faltes d'ortografia com ara la paraula 'dm', que hauria de ser 'damage'. També hi ha un error gramatical en l'estructura de les frases, ja que algunes paraules estan mal col·locades o utilitzades incorrectament.*

- Text: `>els dos partits són ximples > tribalisme estúpid ??`
- Reason: *Possible raó de la mala qualitat: l'ús inadequat dels signes d'interrogació i exclamació.*

- Text: `No tens cor si anomenes \druggies" a la gent que fuma marihuana i he acabat de discutir amb tu, que tinguis un bon dia.`
- Reason: *La frase "No tens cor" es refereix a algú que manca d'empatia o compassió cap als altres. En aquest context, l'autor està expressant ira perquè se sent menyspreat pel fet que l'interlocutor els hagi anomenat "drogat". Per tant, la intenció de les paraules és insultar l'altra persona. Això fa que sigui difícil traduir aquesta part sense perdre matisos importants.*

- Text: `ARGH!!`
- Reason: L*a traducció automàtica d'aquesta paraula anglesa "ARGH" pot donar lloc a diverses paraules catalanes com ara "AAAARG", "AAAAAAAGH", etc., que poden resultar confuses o fins i tot sense sentit en la llengua catalana. Per tant, cal seleccionar acuradament l'opció més adequada segons el context. En aquest cas concret, sembla que es tracta d'una expressió d'enuig, així doncs, la millor opció seria probablement "AAAARG".*

However, the model also exhibited a tendency to be overly strict, sometimes discarding translations that, while informal, were contextually acceptable. This limitation is particularly relevant for social media content, which frequently contains grammatical errors, jargon, and non-standard words. The following are examples of such "false positive" discards:

- Text: `[NOM] Els tatuatges van ser dissenyats específicament per augmentar el seu físic. Aquest noi sembla que la seva germana petita li va fer un gargot mentre dormia.`
- Reason: *La traducció pot tenir errors gramaticals o sintàctics, cosa que afecta negativament la comprensió del missatge original.*

- Text: `Després de veure TLJ, em resulta impossible veure TFA. És una seqüela d'alguna pel·lícula, però aquesta pel·lícula no és TFA.`
- Reason: *La traducció sembla estar feta amb un traductor automàtic que ha generat frases poc naturals o ambigües com ara "és una seqüela d'una pel·lícula", la qual cosa fa difícil entendre si es refereix a una pel·lícula o a alguna altra cosa.*

- Text: `Dóna-li 3 dies, quan tots els punts de venda de MSM es retractin tranquil·lament de la història.`
- Reason: *Possible problemes de traducció: "MSM" pot referir-se a un acrònim o terme que potser no està clar sense context addicional. El nombre "3" també pot ser confús si no queda clar quin tipus d'intervals numèrics estan inclosos aquí. Finalment, l'expressió "retractar-se tranquil·lament" pot necessitar més precisió cultural o idiomàtica perquè sigui clara per als parlants catalans.*

This analysis revealed that while the model could identify egregious translation errors, it struggled with the nuances of informal language, confirming that a fully automated approach was insufficient.

The limitations of automated evaluation required a comprehensive final manual review of the data set to ensure data quality. At the end of the automated process, 50 translations had been discarded. Manual review involved two main tasks. On the one hand, delete bad translations, such as:

- No deagle nerf/vault, bon pegat però podria ser molt millor
- [NOM]
- nah em bo
- ÉS MA'AM!!!!
- Nadó fiddish [NOM]!!
- Canvia' l af̂ôôôôôôôn̂d̂
- 1 bit u/tippr

On the other hand, relabel some texts such as:

- Text: No mantinguis la boca tancada.
- Relabeled sentiment: From Negative to Neutral

- Text: no puc tenir asma
- Relabeled sentiment: From Negative to Neutral

- Text: Al·leluia!!!!!
- Relabeled sentiment: From Neutral to Positive

This manual curation resulted in the removal of approximately 200 additional entries, finalizing the GoEmotions portion of our dataset at 11,638 rows.

4.2.4 Final Dataset

The final step consists of generating the training, validation, and test splits. After combining the previously processed datasets, the rows are shuffled to ensure randomness and reduce potential ordering biases. The data is then divided into three subsets: 80% for training, 10% for validation, and 10% for testing. Finally, the resulting splits are stored as separate CSV files named train, validation, and test.

- The train.csv file contains 19,030 rows distributed as 40.5% Positive, 30.1% Negative and 29.4% Neutral.
- The validation.csv file contains 2,379 rows distributed as 38.1% Positive, 30.4% Negative and 31.4% Neutral.
- The test.csv file contains 2,379 rows distributed as 42.1% Positive, 28.2% Negative and 29.8% Neutral.

4.3 Model Training

With a high-quality dataset prepared, the next phase of this work involves selecting and fine-tuning an appropriate language model. As the primary objective is to perform sentiment classification, an encoder-only architecture has been determined to be the most suitable choice.

As established in Section 3.4, encoder-only models are specifically designed for tasks requiring a deep understanding of input text. The architecture of these models, which leverages the bidirectional context, provides an effective framework for natural language understanding. Prominent examples of this architecture include BERT [49], RoBERTa [107], and ALBERT [106].

For this study, the **roberta-base-ca-v2**[14] model was selected. This is a transformer-based masked language model provided by the Aina Kit, based on the RoBERTa architecture. It was pre-trained on a medium-sized Catalan corpus compiled from various publicly available sources and web crawlers.

Although this model can be used directly for masked language modeling tasks, such as Fill Mask, it is primarily intended for fine-tuning on downstream, non-generative tasks such as Text Classification, Question Answering, or Named Entity Recognition.

The complete implementation of the model training and evaluation process is publicly available as a Jupyter Notebook on the project's GitHub repository[15].

4.3.1 Hyperparameter Tuning

To optimize the performance of the fine-tuning process, a systematic search for the optimal model hyperparameters is necessary. The key hyperparameters tuned in this study are defined as follows:

- **Learning Rate**: A scalar value that controls the step size of the optimizer during training. A carefully chosen learning rate balances the training speed against the risk of instability or failure to converge to an optimal solution.
- **Batch Size**: The number of training examples processed in a single forward and backward pass. This parameter affects the speed of the training and memory consumption and can influence the stability of the learning process.
- **Weight Decay**: A regularization technique that adds a penalty for large weights to the loss function. It is used to mitigate overfitting and improve the model's generalization capabilities on unseen data.

[14] https://huggingface.co/projecte-aina/roberta-base-ca-v2
[15] https://github.com/Danie1Arias/CSXSC/blob/main/train_model.ipynb

- **Number of Epochs**: A single complete pass through the entire training dataset. The number of epochs determines the extent of the model's exposure to the training data, with more epochs potentially improving accuracy at the risk of overfitting.

To identify the best hyperparameter configuration, a grid search methodology is employed. This approach systematically trains and evaluates the model in every possible combination of the specified hyperparameter values to find the configuration that yields the best performance in the validation set.

For this study, the following hyperparameters have been employed: learning rates of $1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}$, batch sizes of 16 and 32, weight decays of 0.01 and 0.05, and training epochs of 4 and 5. In total, 24 possible hyperparameter combinations.

4.3.2 Evaluation Metrics

Evaluation metrics are essential tools for assessing LLMs performance. They provide quantitative insights into how well a model performs in different tasks, measuring aspects such as accuracy, balance between classes, and agreement with ground truth. Furthermore, these metrics help identify common challenges in LLMs, such as hallucinations, biases, or inconsistencies, by providing systematic ways to evaluate performance beyond raw outputs.

In this work, we evaluate the model's performance using four key metrics: *Accuracy*, *F1 Macro*, *F1 Weighted*, and the *Quadratic Weighted Kappa*. Each metric captures different aspects of model behavior and robustness, making them complementary to each other.

Accuracy

Accuracy is the simplest and most widely used metric. It measures the proportion of correct predictions among all predictions made by the model. While intuitive, accuracy may be misleading in cases of class imbalance, as it can be dominated by the majority class.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{4}$$

In binary classification tasks, this formula can also be expressed in terms of the confusion matrix:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives. For multiclass settings, the first (general) definition is used, where the

number of correct predictions corresponds to the total of samples for which the predicted label matches the true label.

F1 Macro

The F1 score is the harmonic mean of precision and recall, providing a balance between these two measures. The *Macro* variant calculates the F1 score independently for each class and then takes their unweighted average. This treats all classes equally, regardless of their frequency, making it particularly useful when evaluating imbalanced datasets.

$$\text{F1}_{\text{macro}} = \frac{1}{C} \sum_{i=1}^{C} \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{6}$$

where C is the total number of classes. Precision and recall for class i are defined as:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}, \quad \text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

F1 Weighted

The *Weighted F1 score* extends Macro F1 by weighting each class's F1 score by its support, for example, the number of true instances of the class. This ensures that larger classes have a greater influence on the final score. Although this metric accounts for the class imbalance, it can still mask poor performance in minority classes.

$$\text{F1}_{\text{weighted}} = \frac{1}{N} \sum_{i=1}^{C} n_i \cdot \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{7}$$

where n_i is the number of true instances of class i, and N is the total number of instances.

Quadratic Weighted Kappa (κ)

The Kappa coefficient, often denoted by κ, measures the level of agreement between two raters. In this case, the model's predictions and the ground-truth labels. Its primary advantage is that it corrects for agreement that could be expected to occur purely by chance. For this study, the *quadratic weighted* variant is used, as it is particularly well-suited for ordinal classification tasks where the classes have a natural order. Unlike metrics such as accuracy, the quadratic weighting scheme penalizes disagreements more heavily when the predicted class is further from the actual class. For example, a prediction of "positive" for a "negative" label receives a larger penalty than a prediction of "neutral."

The coefficient is calculated as follows:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}} \tag{8}$$

where $O_{i,j}$ is the observed agreement matrix (confusion matrix), $E_{i,j}$ is the expected agreement matrix assuming random chance, and $w_{i,j}$ is the quadratic weight matrix that increases with the distance between categories i and j. A κ score of 1 indicates perfect agreement, 0 represents agreement equivalent to random chance, and negative values suggest systematic disagreement.

To provide a qualitative interpretation of the Kappa value, this study refers to the landmark benchmarks proposed by Landis and Koch [187], which has become a widely accepted, though heuristic, standard for interpreting the strength of agreement across many scientific fields. The benchmarks are presented in Table 3.

Table 3: Landis and Koch interpretation scale for the Kappa coefficient.

κ Value	Strength of Agreement
< 0.00	Poor
0.00 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost Perfect

4.3.3 Training and Evaluation Results

Following the hyperparameter tuning strategy outlined previously, a grid search was performed to identify the optimal model configuration. The performance of each configuration was evaluated on the validation set based on the Quadratic Weighted Kappa (κ). The entire training and evaluation process required approximately 11 hours of computation time on a single GPU.

The best-performing configuration was achieved with a learning rate of 2×10^{-5}, a batch size of 32, a weight decay of 0.01, and a training duration of 5 epochs. The epoch-by-epoch results for this optimal configuration are presented in Table 4.

The results in Table 4 demonstrate a clear and positive learning trend. The model achieves its peak performance in the final epoch (Epoch 5), reaching a validation **accuracy of 83.10%** and a κ **score of 0.8667**. A critical observation is the relationship between the training loss and validation loss. While the training loss remains consistently low (around 0.02), the validation loss shows a general downward trend, decreasing to its lowest point of 1.2913 in the final epoch. This pattern is a strong indicator that the model was still generalizing well to unseen data and had not yet begun to overfit, even

after five full epochs. The consistent improvement across all evaluation metrics, including Accuracy, the F1-scores, and κ, further validates the robustness of the training process with the chosen hyperparameters.

Epoch	Tr. Loss	Val. Loss	Accuracy	F1 Macro	F1 Weighted	κ
1	0.0216	1.3458	0.8239	0.8152	0.8242	0.8596
2	0.0249	1.3932	0.8251	0.8138	0.8234	0.8622
3	0.0178	1.4315	0.8268	0.8170	0.8265	0.8596
4	0.0222	1.3137	0.8277	0.8184	0.8279	0.8629
5	**0.0288**	**1.2913**	**0.8310**	**0.8214**	**0.8309**	**0.8667**

Table 4: Performance of the CSXSC model on the validation set using the optimal hyperparameter configuration.

4.4 Model Evaluation and Comparative Analysis

The final phase of this study involves evaluating the performance of the optimized models on the test set. To maximize the amount of training data available for the primary model (CSXSC), the original training and validation datasets were merged. As described in Section 4.4.1, the CSXSC model (based on the RoBERTa encoder-only architecture) was retrained on this aggregated corpus using the optimal hyperparameters identified in the hyperparameter search. Its definitive performance was then assessed on the unseen test dataset.

To contextualize these results, a comparative analysis was conducted with two large-scale, general-purpose LLMs. Specifically, **Starling-LM-7B-alpha** and **Salamandra-7B** were fine-tuned for the same three-class sentiment classification task (see Section 4.4.3). Unlike CSXSC, which follows an encoder-only design tailored to the task, both of these LLMs feature a decoder-only architecture. This enables a direct comparison between a smaller, specialized model and much larger, general-purpose models, highlighting the trade-offs in predictive accuracy, generalization capacity, and computational efficiency.

The complete implementation for this final evaluation and comparative analysis, along with the detailed results, is available in a dedicated Jupyter Notebook within the project's public repository[16].

4.4.1 Regularization Techniques

To ensure that the model generalizes well to unseen data and to prevent overfitting, several regularization and training stabilization techniques were employed. These strategies are crucial to optimize the fine-tuning process.

[16] https://github.com/Danie1Arias/CSXSC/blob/main/evaluate_model.ipynb

Early Stopping

An early stopping strategy has been used to select the most effective version of the model. Although the model trained for a fixed number of cycles (epochs), its performance in the validation dataset is monitored after each cycle. The final model chosen for evaluation was not necessarily the one from the last training cycle, but rather the one that achieved the highest κ score on the validation data. This strategy ensures that the selected model is the one with the best generalization capability, effectively preventing the use of a later version that may have started to overfit.

Dropout

The RoBERTa architecture inherently includes dropout as a form of regularization. During the training process, dropout layers randomly and temporarily deactivate a fraction of neurons in the network for each training example. This technique prevents neurons from becoming overly dependent on each other and forces the model to learn more robust and independent features, thereby enhancing its ability to generalize to new data.

4.4.2 CSXSC Evaluation

The performance of the CSXSC model in the test set is presented in Table 5. The analysis of these results is crucial to identify the optimal model checkpoint and to understand the onset of overfitting.

As highlighted in the table, the model achieves its peak performance at Epoch 4. At this stage, it records the highest scores in all key evaluation metrics: an **Accuracy of 82.43%**, an F1 Macro score of 0.8141, and a **Quadratic Weighted Kappa (κ) of 0.8601**.

This peak marks the point just before the model's generalization capabilities begin to degrade. This is evidenced by the divergence of the loss metrics. While the Training Loss consistently decreases, the Validation Loss begins to increase after Epoch 2. This pattern, coupled with the decline in performance metrics at Epoch 5, confirms that continuing to train beyond the fourth epoch leads to overfitting. Therefore, the model checkpoint from Epoch 4 represents the optimal balance between learning and generalization for this task.

Epoch	Tr. Loss	Val. Loss	Accuracy	F1 Macro	F1 Weighted	κ
1	0.6465	0.4520	0.8134	0.8030	0.8121	0.8438
2	0.3975	0.4481	0.8150	0.8036	0.8124	0.8551
3	0.3141	0.4534	0.8154	0.8034	0.8124	0.8532
4	**0.2550**	**0.4848**	**0.8243**	**0.8141**	**0.8230**	**0.8601**
5	0.2191	0.5063	0.8209	0.8107	0.8199	0.8580

Table 5: Performance of the CSXSC model on the test set using the optimal hyperparameter configuration found in the training stage.

Following the identification of the optimal checkpoint from Epoch 4, the model was evaluated on the held-out test set to assess its final generalization performance. The results of this definitive evaluation are very promising.

The model achieved a final **Accuracy of 83.69%**, an **F1 Macro score of 0.8186**, and a **Quadratic Weighted Kappa (κ) of 0.8715** on the test set. Notably, these scores exceed the peak performance observed on the validation set across all key metrics. This outcome strongly indicates that the model has generalized effectively and did not overfit to the training data. The lower loss and higher accuracy on the completely unseen test set validate the robustness of the selected model and the overall effectiveness of the training and regularization strategy.

In addition to its strong predictive accuracy, the model demonstrated high computational efficiency during evaluation. The entire test set, comprising 2,379 samples, was processed in approximately **14 seconds**. This corresponds to an inference speed of over **170 samples per second**. This high throughput is a significant result, indicating that the CSXSC model is not only accurate but also highly practical for real-world applications where low-latency predictions and efficient use of computational resources are required. The combination of high accuracy and fast inference speed validates the model as an effective and deployable solution for Catalan sentiment analysis.

4.4.3 Optimization Techniques

Now that we have established the performance of the CSXSC model, our aim is to compare it with other general-purpose LLMs. These models are computationally heavier given that they contain billions of parameters, in contrast to the millions in the CSXSC model. This vast number of parameters demands significant memory (VRAM) and processing power, making standard fine-tuning infeasible on consumer-grade hardware.

Therefore, to fine-tune these large models and evaluate them on our test set, it was necessary to apply a suite of advanced optimization techniques. These methods are designed to drastically reduce the memory footprint of the model during training without significant performance degradation.

QLoRA

This is the primary optimization strategy used, which combines two techniques:

- **4-bit Quantization:** The core of this method involves reducing the precision of the model weights. The billions of parameters, typically stored as 32-bit floating-point numbers, are compressed into a much smaller 4-bit data type. This is conceptually similar to reducing the file size of a high-resolution image. It dramatically shrinks the model's size in memory, making it possible to load onto the GPU.
- **Low-Rank Adaptation (LoRA):** Instead of fine-tuning all billions of the model's parameters, LoRA freezes the entire pre-trained model and injects a very small num-

ber of new, trainable "adapter" layers. During training, only these tiny adapters (often less than 1% of the total size of the model) are updated. This drastically reduces the memory required to store gradients and optimizer states, which is the most memory-intensive part of the training process.

BFloat16 Mixed-Precision Training

This technique leverages the fact that not all calculations during training require full 32-bit precision. By performing many of the operations using the 16-bit BFloat16 format, the memory required for storing activations is halved, and the overall training process is significantly accelerated on compatible GPUs.

Paged Optimizers

The optimizer itself, which calculates the weight updates, is a major consumer of VRAM. An 8-bit paged optimizer is a memory-efficient variant that uses a technique called "paging" to intelligently transfer optimizer states from the GPU to the main system RAM when they are not in use, thus preventing out-of-memory errors.

Gradient Accumulation

This technique allows the model to simulate the effects of a large batch size without the corresponding high memory cost. The model processes several smaller "micro-batches" sequentially and accumulates their gradients. Only after a specified number of these steps are the accumulated gradients used to update the model's weights. This maintains the training stability of a large batch size while only requiring enough VRAM to hold a single small batch.

4.4.4 Starling-LM-7B-alpha Evaluation

The first large-scale model used for comparison is Starling-LM-7B-alpha[17], a 7-billion-parameter, decoder-only model. As described by its developers, Starling is not a base model, but rather an instruction-tuned language model fine-tuned using Reinforcement Learning from AI Feedback (RLAIF). Its primary training objective was to improve its capabilities as a helpful and harmless chat assistant. This is a critical distinction, as its architecture and training are optimized for generative, conversational tasks rather than discriminative classification.

The model was fine-tuned on our sentiment classification task using the QLoRA optimization technique. The performance in the validation set is detailed in Table 6. The results show a clear learning trend, with performance improving across all metrics throughout the three training epochs. The model achieved its peak performance on the validation set at Epoch 3, with an accuracy of 69.31% and a κ of 0.7055. It is worth noting that the training loss, while decreasing, remained significantly high (6.6027), which suggests

[17] https://huggingface.co/berkeley-nest/Starling-LM-7B-alpha

that the sentiment classification task is inherently difficult for this model's generative architecture.

Epoch	Tr. Loss	Val. Loss	Accuracy	F1 Macro	F1 Weighted	κ
1	14.4836	1.2961	0.6566	0.6485	0.6612	0.6527
2	8.1661	1.1288	0.6906	0.6700	0.6829	0.7093
3	6.6027	1.0398	0.6931	0.6789	0.6916	0.7055

Table 6: Performance of the Starling-LM-7B-alpha model.

Following the training, the best-performing checkpoint from Epoch 3 was evaluated in the holdout test set. The model achieved a final **Accuracy of 68.39%**, an **F1 Macro score of 0.6489**, and a **Quadratic Weighted Kappa (κ) of 0.7107** on the test set. These final scores are consistent with the performance on the validation set, indicating that the model did not overfit. However, they are substantially lower than those achieved by the specialized CSXSC model.

Furthermore, the computational cost was significant. The evaluation on the test set alone took approximately 391 seconds (more than 6.5 minutes), with an inference speed of only 6 samples per second. This is a stark contrast to the CSXSC model's 170 samples per second, highlighting the major trade-off in computational efficiency when using these large-scale generative models for classification tasks.

4.4.5 Salamandra-7B Evaluation

The second large-scale model evaluated is Salamandra-7B[18], a 7-billion-parameter, decoder-only language model based on the Llama 2 architecture. This model was specifically chosen for its extensive pre-training on a large, high-quality corpus that includes a significant amount of Catalan text. The hypothesis for this experiment was that a model with a stronger foundational understanding of the Catalan language might achieve better performance on this specific downstream task, even when compared to a highly capable instruction-tuned model like Starling.

The model was fine-tuned on our sentiment classification task using the same QLoRA optimization technique. The performance on the validation set is detailed in Table 7. The validation results show a consistent improvement in performance, with the model reaching its peak at Epoch 3. At this point, it achieved an accuracy of 74.69% and a κ of 0.7601. As with the Starling model, the training loss remained high, though it was considerably lower (4.0926), suggesting that Salamandra's architecture and pre-training were a better fit for the classification task.

[18] https://huggingface.co/BSC-LT/salamandra-7b

Epoch	Tr. Loss	Val. Loss	Accuracy	F1 Macro	F1 Weighted	κ
1	5.9020	0.5874	0.7411	0.7266	0.7381	0.7451
2	4.4102	0.5632	0.7457	0.7322	0.7434	0.7644
3	4.0926	0.5559	0.7469	0.7316	0.7431	0.7601

Table 7: Performance of the Salamandra-7B model.

Following the training, the best-performing checkpoint from Epoch 3 was evaluated on the holdout test set. The model achieved a final **Accuracy of 75.33%**, an **F1 Macro score of 0.7301**, and a **Quadratic Weighted Kappa (κ) of 0.7872** on the test set. These results confirm the hypothesis: the Salamandra model, with its extensive Catalan pre-training, significantly outperformed the Starling model. However, its performance still falls short of that achieved by the specialized CSXSC model.

In terms of computational cost, the evaluation in the test set took approximately 333 seconds (about 5.5 minutes), with an inference speed of 7 samples per second. Although this is slightly faster than the Starling model, it is still orders of magnitude slower than the CSXSC model, further reinforcing the efficiency advantage of using a smaller, specialized architecture for this task.

4.4.6 Comparative Analysis

This section presents a direct comparison of the final performance and computational efficiency of the three evaluated models on the held-out test set. A summary of the key results is provided in Table 8, which contrasts the specialized CSXSC model with the two large-scale, general-purpose LLMs.

Model	Acc. (%)	κ	F1 Macro	F1 Weighted	Speed
CSXSC	**83.69**	**0.8715**	**0.8186**	**0.8364**	**∼170**
Salamandra-7B	75.33	0.7872	0.7301	0.7512	∼7
Starling-LM-7B-alpha	68.39	0.7107	0.6489	0.6767	∼6

Table 8: Final performance and efficiency comparison of the three models on the test set. Speed is measured in samples per second.

The results presented in the table clearly delineate the performance hierarchy among the models. The CSXSC model, based on the RoBERTa-base architecture, achieved the highest scores across all predictive metrics, including an F1 Macro of 0.8186 and a κ of 0.8715. Of the two 7-billion-parameter models, Salamandra-7B registered a stronger performance, surpassing Starling-LM-7B-alpha by a notable margin across all reported metrics.

In terms of computational efficiency, a significant disparity is observed. The smaller CSXSC model demonstrated an inference speed of approximately 170 samples per second, which is more than **24 times faster** than the two larger, decoder-only models.

This highlights the substantial trade-off between model size, predictive performance, and computational requirements for this specific task.

4.5 Conclusions

This work successfully developed and evaluated a specialized sentiment classification model for Catalan social media (CSXSC) and benchmarked its performance against two large-scale, general-purpose language models. The comprehensive process of dataset creation, model training, and comparative analysis has yielded several key conclusions that are critical for the practical application of LLMs in specific domains and languages.

- **Architectural suitability outweighs model size for classification tasks.** The most significant finding of this study is that the smaller, ∼125M-parameter, encoder-only CSXSC model **significantly outperformed** both 7-billion-parameter, decoder-only models. With a final accuracy of 83.69% and a κ of 0.8715, the CSXSC model demonstrated that for discriminative, natural language understanding tasks like sentiment analysis, choosing an architecture specifically designed for the task is more critical than the raw scale of the model.
- **Specialized models offer vastly superior computational efficiency.** The trade-off between performance and cost was a central theme. The CSXSC model was not only more accurate but also drastically more efficient, with an inference speed of approximately 170 samples per second. This was over **24 times faster** than the larger models, which processed only 6-7 samples per second. This finding underscores the practical advantage of using smaller, specialized models for real-world deployment where low latency and cost-effectiveness are crucial.
- **Language-specific Pre-training enhances fine-tuning performance.** The comparison between the two 7B models yielded a clear conclusion: the **Salamandra-7B** model, which was pre-trained on a substantial Catalan corpus, achieved significantly higher accuracy (75.33%) than the more general, instruction-tuned **Starling-LM-7B-alpha** (68.39%). This demonstrates that a strong foundational knowledge of the target language's specific vocabulary and nuances provides a better starting point for fine-tuning on downstream tasks.
- **Meticulous dataset curation is foundational for reliable results.** The initial phase of this work, which involved aggregating three distinct datasets, balancing the class distribution, and implementing a multi-step quality filtering process (including both automated and manual curation), was a prerequisite for success. This process highlights that the performance of any fine-tuned model is fundamentally dependent on the quality and representativeness of its training data.
- **Advanced optimization makes large model research accessible.** This study successfully demonstrated the effectiveness of modern optimization techniques. By employing **QLoRA** (4-bit Quantization and Low-Rank Adaptation) in conjunction with other methods like paged optimizers and mixed-precision training, it was

possible to fine-tune and evaluate 7-billion-parameter models on consumer-grade hardware. This confirms that such advanced comparative studies are no longer exclusively the domain of large, resource-rich institutions.

4.6 Future Work and Directions

While this study successfully developed a robust sentiment classification model and provided a clear comparative analysis, it also opens up several promising avenues for future research. The findings and limitations of this work suggest the following directions:

- **Scaling with Advanced Computational Resources:** This project was conducted using open-source models and freely available tools on consumer-grade hardware. A logical next step would be to explore the performance of top-tier, proprietary models (such as the GPT and Claude families) via paid APIs. Furthermore, training even larger open-source models (e.g., 30B+ parameters) on high-performance cloud infrastructure (such as NVIDIA A100 or H100 GPUs) could reveal new insights into the relationship between scale and task-specific performance.
- **Automated Dataset Refinement:** The current dataset, while carefully curated, still contains the noisy, informal language characteristic of social media. Future work could involve using a powerful generative model to "clean" or "normalize" the dataset texts—correcting grammar, expanding slang, and standardizing expressions. Training the models on this refined dataset could reveal the precise impact of text quality on sentiment classification performance.
- **Agentic and Chain-of-Thought Classification:** The rise of agentic AI systems presents a new paradigm for classification. Instead of a simple input-output prediction, future research could explore multi-step reasoning approaches. An LLM agent could be prompted to first analyze and "reason" about the sentiment in a review (e.g., using a Chain-of-Thought process to list pros and cons) before assigning a final label. This could significantly improve accuracy on complex or ambiguous texts.
- **Deeper Domain Specialization for Large Models:** This study showed that language-specific pre-training (Salamandra) improved performance. A powerful extension would be to take a large base model like Salamandra-7B and perform *continued pre-training* on a massive, newly scraped corpus of Catalan social media text. After this domain-adaptation step, the model could then be fine-tuned for sentiment analysis, potentially creating a large model that combines the best of both worlds: scale and deep domain expertise.
- **Fine-Grained Emotion Classification:** This work focused on the three broad categories of positive, negative, and neutral sentiment. However, the GoEmotions dataset contains 27 distinct emotion labels. A compelling future project would be to leverage this rich data to develop a model for fine-grained, multi-label *emotion*

classification in Catalan, capable of identifying nuanced feelings like "admiration," "annoyance," or "gratitude" rather than just sentiment polarity.

4.7 Ethical Considerations

The development and deployment of any artificial intelligence system, including the sentiment analysis models presented in this work, carry significant ethical responsibilities. While this research focused on the technical challenges of creating a high-performance classifier for Catalan, it is crucial to acknowledge the broader societal context in which such a tool could be used.

A primary consideration is the potential for bias embedded within the model. The datasets used for training, sourced from restaurant reviews and social media platforms like Reddit, reflect the demographic, cultural, and linguistic biases of their users. Consequently, the model may learn to unfairly associate certain dialects, slang, or cultural expressions with a particular sentiment, potentially leading to the misrepresentation of opinions from specific user groups. An ongoing audit for fairness and bias is essential to ensure the model does not systematically disadvantage any segment of the population.

Furthermore, the use of public social media data raises questions of privacy and user consent. Although the data in this study was sourced from publicly available and anonymized datasets, the practice of training models on user-generated content must be approached with care. Users may not be fully aware that their public expressions are being used to build and train AI systems, and the purpose of this data collection should be transparent.

Finally, the potential for misuse of this technology must be addressed. A reliable sentiment classifier could be employed for beneficial purposes, such as helping businesses understand customer feedback or enabling social scientists to gauge public opinion. However, it could also be used for ethically questionable applications. These include the automated surveillance of public discourse to suppress dissent, the targeted manipulation of emotionally vulnerable individuals for commercial or political gain, or the implementation of biased content moderation systems that unfairly censor legitimate criticism under the guise of "negativity." Therefore, the deployment of the CSXSC model, or any similar technology, must be accompanied by a strong ethical framework, transparency regarding its capabilities and limitations, and clear lines of accountability for its impact.

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Dr. Jordi Pascual Fontanilles. This work and its results would not have been possible without his invaluable guidance and unwavering support. The journey of defining, investigating, and executing this thesis over the past six months has been a significant undertaking, particularly as it ventured into topics like Natural Language Processing that extend beyond the core curriculum of the Master's in Computer Security Engineering and Artificial Intelligence. Dr. Pascual's mentorship was instrumental in navigating this challenge, always providing clarity when problems occurred and fostering an environment of collaborative discovery. I am sincerely thankful for his dedication.

Last, but certainly not least, I wish to extend my thanks to the entire faculty of the master's program. It has been a stimulating and enriching year at the Universitat Rovira i Virgili, and I am grateful to have had the opportunity to learn from such experienced and reputable professors from around the world.

References

[1] K. S. Henry Brighton and S. Kirby, "Language as an evolutionary system," *Physics of Life Reviews*, pp. 177–226, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571064505000229

[2] J. D. Wright, Ed., *International Encyclopedia of the Social and Behavioral Sciences*. Oxford: Elsevier, 2015. [Online]. Available: https://shop.elsevier.com/books/international-encyclopedia-of-the-social-and-behavioral-sciences/wright/978-0-08-097086-8

[3] P. Lieberman, "Language did not spring forth 100,000 years ago," *PLOS Biology*, 2015.

[4] SIL International, "How many languages are there in the world?" 2024, accessed: 2025-03-25. [Online]. Available: https://www.ethnologue.com/insights/how-many-languages/

[5] N. Isern and J. Fort, "Language extinction and linguistic fronts," *Journal of The Royal Society Interface*, 2014. [Online]. Available: https://www.researchgate.net/publication/260561799_Language_extinction_and_linguistic_fronts

[6] A. M. Turing, "Computing machinery and intelligence," *Mind*, pp. 433–460, 1950. [Online]. Available: https://academic.oup.com/mind/article/LIX/236/433/986238

[7] R. M. French, "The turing test: 50 years later," *Trends in Cognitive Sciences*, 2000. [Online]. Available: https://www.researchgate.net/publication/2435828_Turing_test_50_years_later

[8] H. N. et al., "A comprehensive overview of large language models," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2307.06435

[9] M. Gams and S. Kramar, "Evaluating chatgpt's consciousness and its capability to pass the turing test: A comprehensive analysis," *Journal of Computer and Communications*, pp. 219–237, 2024. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=132146

[10] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman, "Gpqa: A graduate-level google-proof qa benchmark," 2023. [Online]. Available: https://arxiv.org/abs/2311.12022

[11] I. University, "Understand measures of supercomputer performance and storage system capacity," 2024, accessed: 2025-04-11. [Online]. Available: https://servicenow.iu.edu/kb?id=kb_article_view&sysparm_article=KB0023145

[12] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

[13] K. Verspoor and K. Cohen, *Natural Language Processing*, 01 2013, pp. 1495–1498.

[14] R. C. Schank, *Conceptual Information Processing*. New York: North-Holland, 1977.

[15] J. R. Searle, "Minds, brains, and programs," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417–424, 1980.

[16] M. Awad and R. Khanna, *Hidden Markov Model*, 01 2015, pp. 81–104.

[17] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, p. 282–289.

[18] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[22] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," 2021. [Online]. Available: https://arxiv.org/abs/2106.07139

[23] Supriyono, A. P. Wibawa, Suyono, and F. Kurniawan, "Advancements in natural language processing: Implications, challenges, and future directions," *Telematics and Informatics Reports*, vol. 16, p. 100173, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772503024000598

[24] S. Kumar, P. P. Roy, D. P. Dogra, and B.-G. Kim, "A comprehensive review on sentiment analysis: Tasks, approaches and applications," 2023. [Online]. Available: https://arxiv.org/abs/2311.11250

[25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. A. et al., "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[26] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, and A. K. et al., "Lamda: Language models for dialog applications," 2022. [Online]. Available: https://arxiv.org/abs/2201.08239

[27] B. Workshop, :, T. L. Scao, A. Fan, C. Akiki, E. Pavlick, and S. I. et al., "Bloom: A 176b-parameter open-access multilingual language model," 2023. [Online]. Available: https://arxiv.org/abs/2211.05100

[28] J. R. Jim, M. A. R. Talukder, P. Malakar, M. M. Kabir, K. Nur, and M. Mridha, "Recent advancements and challenges of nlp-based sentiment analysis: A state-of-the-art review," *Natural Language Processing Journal*, vol. 6, p. 100059, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949719124000074

[29] M. Rodríguez-Ibánez, A. Casánez-Ventura, F. Castejón-Mateos, and P.-M. Cuenca-Jiménez, "A review on sentiment analysis from social media platforms," *Expert Systems with Applications*, vol. 223, p. 119862, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423003639

[30] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[31] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[34] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[35] K. L. Tan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim, "Roberta-lstm: a hybrid model for sentiment analysis with transformer and recurrent neural network," *IEEE Access*, vol. 10, pp. 21 517–21 525, 2022.

[36] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap, "Relational recurrent neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[37] S. Yu, S. R. Indurthi, S. Back, and H. Lee, "A multi-stage memory augmented neural network for machine reading comprehension," in *Proceedings of the workshop on machine reading for question answering*, 2018, pp. 21–30.

[38] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," *arXiv preprint arXiv:1410.8206*, 2014.

[39] G. Wiese, D. Weissenborn, and M. Neves, "Neural domain adaptation for biomedical question answering," *arXiv preprint arXiv:1706.03610*, 2017.

[40] R. Newatia, "How to implement cnn for nlp tasks like sentence classification," 2019, accessed: 2025-04-18. [Online]. Available: https://medium.com/saarthi-ai/sentence-classification-using-convolutional-neural-networks-ddad72c7048c

[41] W. Wang and J. Gang, "Application of convolutional neural network in natural language processing," in *2018 international conference on information Systems and computer aided education (ICISCAE)*. IEEE, 2018, pp. 64–70.

[42] J. Xiao and Z. Zhou, "Research progress of rnn language model," in *2020 ieee international conference on artificial intelligence and computer applications (icaica)*. IEEE, 2020, pp. 1285–1288.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[46] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[47] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," *arXiv preprint arXiv:1911.05507*, 2019.

[48] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020.

[49] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[50] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimedia Tools and Applications*, vol. 82, no. 3, p. 3713–3744, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1007/s11042-022-13428-4

[51] X. Song, A. Salcianu, Y. Song, D. Dopson, and D. Zhou, "Fast wordpiece tokenization," 2021. [Online]. Available: https://arxiv.org/abs/2012.15524

[52] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[53] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, 01 2014, pp. 1532–1543.

[54] H. Schmid, "Part-of-speech tagging with neural networks," *arXiv preprint cmp-lg/9410018*, 1994.

[55] B. Mohit, "Named entity recognition," in *Natural language processing of semitic languages*. Springer, 2014, pp. 221–245.

[56] R. P. Van Gompel and M. J. Pickering, "Syntactic parsing," *The Oxford handbook of psycholinguistics*, pp. 289–307, 2007.

[57] R. Lass, *Phonology: An introduction to basic concepts*. Cambridge University Press, 1984.

[58] A. Carstairs-McCarthy, *Introduction to English Morphology: words and their structure*. Edinburgh university press, 2017.

[59] D. Jurafsky, *Speech & language processing*. Pearson Education India, 2000.

[60] E. Agirre and P. Edmonds, *Word sense disambiguation: Algorithms and applications*. Springer Science & Business Media, 2007, vol. 33.

[61] S. C. Levinson, "Pragmatics," *Cambridge UP*, 1983.

[62] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, and Łukasz Kaiser et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016. [Online]. Available: https://arxiv.org/abs/1609.08144

[63] C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf, "Accelerated dp based search for statistical translation." in *Eurospeech*, 1997, pp. 2667–2670.

[64] S. Bangalore, O. Rambow, and S. Whittaker, "Evaluation metrics for generation," in *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, 2000, pp. 1–8.

[65] S. Nießen, F. J. Och, G. Leusch, H. Ney *et al.*, "An evaluation tool for machine translation: Fast evaluation for mt research." in *LREC*, 2000.

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[67] G. Doddington, "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics," in *Proceedings of the second international conference on Human Language Technology Research*, 2002, pp. 138–145.

[68] P. J. Hayes, "Intelligent high-volume text processing using shallow, domain-specific techniques," in *Text-based intelligent systems*. Psychology Press, 2014, pp. 227–241.

[69] W. W. Cohen *et al.*, "Learning rules that classify e-mail," in *AAAI spring symposium on machine learning in information access*, vol. 18. California, 1996, p. 25.

[70] J. Rennie, "ifile: An application of machine learning to e-mail filtering," in *Proc. KDD 2000 Workshop on text mining, Boston, MA*. Citeseer, 2000.

[71] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos, "A memory-based approach to anti-spam filtering for mailing lists," *Information retrieval*, vol. 6, pp. 49–73, 2003.

[72] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[73] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," *arXiv preprint cs/0109015*, 2001.

[74] A. Berger, S. A. Della Pietra, and V. J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[75] T. Xia, "A constant time complexity spam detection algorithm for boosting throughput on rule-based filtering systems," *IEEE Access*, vol. 8, pp. 82 653–82 661, 2020.

[76] E. Morin, "Automatic acquisition of semantic relations between terms from technical corpora," in *Proc. of the Fifth International Congress on Terminology and Knowledge Engineering-TKE'99*, 1999.

[77] N. Bondale, P. Maloor, A. Vaidyanathan, S. Sengupta, and P. Rao, "Extraction of information from open-ended questionnaires using natural language processing techniques," *Computer Science and Informatics*, vol. 29, no. 2, pp. 15–22, 1999.

[78] D. M. Zajic, B. J. Dorr, and J. Lin, "Single-document and multi-document summarization techniques for email threads using sentence compression," *Information Processing & Management*, vol. 44, no. 4, pp. 1600–1610, 2008.

[79] D. Wang, S. Zhu, T. Li, and Y. Gong, "Multi-document summarization using sentence-based topic models," in *Proceedings of the ACL-IJCNLP 2009 conference short papers*, 2009, pp. 297–300.

[80] D. Wang, S. Zhu, T. Li, Y. Chi, and Y. Gong, "Integrating document clustering and multidocument summarization," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 3, pp. 1–26, 2011.

[81] H. Fang, W. Lu, F. Wu, Y. Zhang, X. Shang, J. Shao, and Y. Zhuang, "Topic aspect-oriented summarization via group selection," *Neurocomputing*, vol. 149, pp. 1613–1619, 2015.

[82] E. D. Liddy, "Natural language processing," 2001.

[83] N. Sager, M. Lyman, N. T. Nhan, and L. J. Tick, "Medical language processing: applications to patient data representation and automatic encoding," *Methods of information in medicine*, vol. 34, no. 01/02, pp. 140–146, 1995.

[84] M. Lyman, N. Sager, C. Friedman, and E. Chi, "Computer-structured narrative in ambulatory care: its use in longitudinal review of clinical data," in *proceedings of the annual symposium on computer application in medical care*, 1985, p. 82.

[85] A. T. McCray, S. Srinivasan, and A. C. Browne, "Lexical methods for managing variation in biomedical terminologies," in *proceedings of the annual symposium on computer application in medical care*, 1994, p. 235.

[86] C. Friedman, J. J. Cimino, and S. B. Johnson, "A conceptual model for clinical radiology reports," in *proceedings of the annual symposium on computer application in medical care*, 1993, p. 829.

[87] Z. Wang, Z. Chu, T. V. Doan, S. Ni, M. Yang, and W. Zhang, "History, development, and principles of large language models-an introductory survey," 2024. [Online]. Available: https://arxiv.org/abs/2402.06853

[88] F. Jelinek, *Statistical methods for speech recognition*. MIT press, 1998.

[89] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI*, 2019, accessed: 2024-11-15. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[90] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, and S. A. et al., "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[91] Y. Yuhang, P. Yizhou, E. Chng, and X. Zhong, "Bridging speech and text: Enhancing asr with pinyin-to-character pre-training in llms," 09 2024.

[92] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[93] J. L. Gastaldi, J. Terilla, L. Malagutti, B. DuSell, T. Vieira, and R. Cotterell, "The foundations of tokenization: Statistical and computational concerns," 2025. [Online]. Available: https://arxiv.org/abs/2407.11606

[94] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2025, online manuscript released January 12, 2025. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[95] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://aclanthology.org/P16-1162/

[96] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," 2002. [Online]. Available: https://arxiv.org/abs/cs/0205057

[97] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, S. Ananiadou, Ed. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 177–180. [Online]. Available: https://aclanthology.org/P07-2045/

[98] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. R. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. S. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *ArXiv*, vol. abs/1609.08144, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:3603249

[99] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 66–75. [Online]. Available: https://aclanthology.org/P18-1007/

[100] Y. Zhao, W. Zhang, G. Chen, K. Kawaguchi, and L. Bing, "How do large language models handle multilingualism?" 2024. [Online]. Available: https://arxiv.org/abs/2402.18815

[101] K. Cao and L. Rimell, "You should evaluate your language model on marginal likelihood over tokenisations," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and

S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2104–2114. [Online]. Available: https://aclanthology.org/2021.emnlp-main.161/

[102] C. Wang, M. Li, and A. J. Smola, "Language models with transformers," 2019. [Online]. Available: https://arxiv.org/abs/1904.09408

[103] T. Wang, A. Roberts, D. Hesslow, T. L. Scao, H. W. Chung, I. Beltagy, J. Launay, and C. Raffel, "What language model architecture and pretraining objective work best for zero-shot generalization?" 2022. [Online]. Available: https://arxiv.org/abs/2204.05832

[104] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," 2021. [Online]. Available: https://arxiv.org/abs/2006.03654

[105] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," 2020. [Online]. Available: https://arxiv.org/abs/2003.10555

[106] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," 2020. [Online]. Available: https://arxiv.org/abs/1909.11942

[107] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[108] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," 2020. [Online]. Available: https://arxiv.org/abs/1911.02116

[109] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683

[110] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, and T. L. S. et al., "Multitask prompted training enables zero-shot task generalization," 2022. [Online]. Available: https://arxiv.org/abs/2110.08207

[111] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, J. Kernion, and K. N. et al., "A general language assistant as a laboratory for alignment," 2021. [Online]. Available: https://arxiv.org/abs/2112.00861

[112] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, and H. et al., "Competition-level code generation with alphacode," *Science*, vol. 378, no. 6624, p. 1092–1097, Dec. 2022. [Online]. Available: http://dx.doi.org/10.1126/science.abq1158

[113] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, and G. B. et al., "Evaluating large language models trained on code," 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[114] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, and A. H. et al., "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[115] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, and P. G. S. et al., "Gemma: Open models based on gemini research and technology," 2024. [Online]. Available: https://arxiv.org/abs/2403.08295

[116] M. Eriksson, E. Purificato, A. Noroozian, J. Vinagre, G. Chaslot, E. Gomez, and D. Fernandez-Llorca, "Can we trust ai benchmarks? an interdisciplinary review of current issues in ai evaluation," 2025. [Online]. Available: https://arxiv.org/abs/2502.06559

[117] R. Pfister and H. Jud, "Understanding and benchmarking artificial intelligence: Openai's o3 is not agi," 2025. [Online]. Available: https://arxiv.org/abs/2501.07458

[118] I. D. Raji, E. Denton, E. M. Bender, A. Hanna, and A. Paullada, "AI and the everything in the whole wide world benchmark," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=j6NxpQbREA1

[119] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. III, and K. Crawford, "Datasheets for datasets," 2021. [Online]. Available: https://arxiv.org/abs/1803.09010

[120] R. Denton, A. Hanna, R. Amironesei, A. Smart, H. Nicole, and M. K. Scheuerman, "Bringing the people back in: Contesting benchmark machine learning datasets," 2020. [Online]. Available: https://arxiv.org/abs/2007.07399

[121] V. Arzt and A. Hanbury, "Beyond the numbers: Transparency in relation extraction benchmark creation and leaderboards," 2024. [Online]. Available: https://arxiv.org/abs/2411.05224

[122] J. Michael, A. Holtzman, A. Parrish, A. Mueller, A. Wang, A. Chen, D. Madaan, N. Nangia, R. Y. Pang, J. Phang, and S. R. Bowman, "What do nlp researchers believe? results of the nlp community metasurvey," 2022. [Online]. Available: https://arxiv.org/abs/2208.12852

[123] E. Gómez, L. Porcaro, P. Frau, and J. Vinagre, *Diversity in artificial intelligence conferences – An analysis of indicators for gender, country and institution diversity from 2007 to 2023*. Publications Office of the European Union, 2024.

[124] M. Rauh, N. Marchal, A. Manzini, L. A. Hendricks, R. Comanescu, C. Akbulut, T. Stepleton, J. Mateos-Garcia, S. Bergman, J. Kay, C. Griffin, B. Bariach, I. Gabriel, V. Rieser, W. Isaac, and L. Weidinger, "Gaps in the safety evaluation of generative ai," *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, vol. 7, no. 1, pp. 1200–1217, Oct. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AIES/article/view/31717

[125] G. Grill, "Constructing capabilities: The politics of testing infrastructures for generative ai," in *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1838–1849. [Online]. Available: https://doi.org/10.1145/3630106.3659009

[126] N. Alzahrani, H. A. Alyahya, Y. Alnumay, S. Alrashed, S. Alsubaie, Y. Almushaykeh, F. Mirza, N. Alotaibi, N. Altwairesh, A. Alowisheq, M. S. Bari, and H. Khan, "When benchmarks are targets: Revealing the sensitivity of large language model leaderboards," 2024. [Online]. Available: https://arxiv.org/abs/2402.01781

[127] W. Orr and E. B. Kang, "Ai as a sport: On the competitive epistemologies of benchmarking," in *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1875–1884. [Online]. Available: https://doi.org/10.1145/3630106.3659012

[128] E. Denton, A. Hanna, R. Amironesei, A. Smart, and H. Nicole, "On the genealogy of machine learning datasets: A critical history of imagenet," *Big Data & Society*, vol. 8, no. 2, p. 20539517211035955, 2021. [Online]. Available: https://doi.org/10.1177/20539517211035955

[129] S. Biderman, H. Schoelkopf, L. Sutawika, L. Gao, J. Tow, B. Abbasi, A. F. Aji, P. S. Ammanamanchi, S. Black, J. Clive, A. DiPofi, J. Etxaniz, B. Fattori, J. Z. Forde, C. Foster, J. Hsu, M. Jaiswal, W. Y. Lee, H. Li, C. Lovering, N. Muennighoff, E. Pavlick, J. Phang, A. Skowron, S. Tan, X. Tang, K. A. Wang, G. I. Winata, F. Yvon, and A. Zou, "Lessons from the trenches on reproducible evaluation of language models," 2024. [Online]. Available: https://arxiv.org/abs/2405.14782

[130] S. Ott, A. Barbosa-Silva, K. Blagec, J. Brauner, and M. Samwald, "Mapping global dynamics of benchmark creation and saturation in artificial intelligence," *Nature Communications*, vol. 13, no. 1, 2022. [Online]. Available: https://doi.org/10.1038/s41467-022-34591-0

[131] T. R. McIntosh, T. Susnjak, N. Arachchilage, T. Liu, D. Xu, P. Watters, and M. N. Halgamuge, "Inadequacies of large language model benchmarks in the era of generative artificial intelligence," *IEEE Transactions on Artificial Intelligence*, p. 1–18, 2025. [Online]. Available: http://dx.doi.org/10.1109/TAI.2025.3569516

[132] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," 2023. [Online]. Available: https://arxiv.org/abs/2311.17035

[133] B. Steenhoek, M. M. Rahman, R. Jiles, and W. Le, "An empirical study of deep learning models for vulnerability detection," 2023. [Online]. Available: https://arxiv.org/abs/2212.08109

[134] Z. Li, S. Lu, D. Guo, N. Duan, S. Jannu, G. Jenks, D. Majumder, J. Green, A. Svyatkovskiy, S. Fu, and N. Sundaresan, "Automating code review activities by large-scale pre-training," 2022. [Online]. Available: https://arxiv.org/abs/2203.09095

[135] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified pre-training for program understanding and generation," 2021. [Online]. Available: https://arxiv.org/abs/2103.06333

[136] J. He, Z. Xin, B. Xu, T. Zhang, K. Kim, Z. Yang, F. Thung, I. Irsan, and D. Lo, "Representation learning for stack overflow posts: How far are we?" 2024. [Online]. Available: https://arxiv.org/abs/2303.06853

[137] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W. tau Yih, L. Zettlemoyer, and M. Lewis, "Incoder: A generative model for code infilling and synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2204.05999

[138] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, and T. R. et al., "Code llama: Open foundation models for code," 2024. [Online]. Available: https://arxiv.org/abs/2308.12950

[139] H. Le, Y. Wang, A. D. Gotmare, S. Savarese, and S. C. H. Hoi, "Coderl: Mastering code generation through pretrained models and deep reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2207.01780

[140] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2203.13474

[141] G. McKenna, "Over 25% of google's code is now written by ai—and ceo sundar pichai says it's just the start," *Fortune*, 2024, accessed: 2025-07-16. [Online]. Available: https://fortune.com/2024/10/30/googles-code-ai-sundar-pichai/

[142] A. V. Sadybekov and V. Katritch, "Computational approaches streamlining drug discovery," *Nature*, 2023. [Online]. Available: https://www.nature.com/articles/s41586-023-05905-z

[143] N. Savage, "Drug discovery companies are customizing chatgpt: here's how," *Nature*, 2023. [Online]. Available: https://www.nature.com/articles/s41587-023-01788-7

[144] B. Haley and F. Roudnicky, "Functional genomics for cancer drug target discovery," *Cancer Cell*, vol. 38, no. 1, pp. 31–43, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1535610820302051

[145] Z. Zhang, S. Zohren, and S. Roberts, "Deep learning for portfolio optimization," *The Journal of Financial Data Science*, vol. 2, no. 4, p. 8–20, Aug. 2020. [Online]. Available: http://dx.doi.org/10.3905/jfds.2020.1.042

[146] A. Mashrur, W. Luo, N. Zaidi, and A. Robles-Kelly, "Machine learning for financial risk management: A survey," *IEEE Access*, vol. 8, pp. 203 203–203 223, 01 2020.

[147] C. Pagliaro, D. Mehta, H.-T. Shiao, S. Wang, and L. Xiong, "Investor behavior modeling by analyzing financial advisor notes: A machine learning perspective," 2021. [Online]. Available: https://arxiv.org/abs/2107.05592

[148] A. Shah, P. Raj, P. Kumar, S. P, and A. V, "Finaid, a financial advisor application using ai," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, pp. 2282–2286, 05 2020.

[149] C. V. Misischia, F. Poecze, and C. Strauss, "Chatbots in customer service: Their relevance and impact on service quality," *Procedia Computer Science*, vol. 201, pp. 421–428, 2022, the 13th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 5th International Conference on Emerging Data and Industry 4.0 (EDI40). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922004689

[150] "Chatbots in consumer finance," *Consumer Financial Protection Bureau*, 2023, accessed: 2025-07-16. [Online]. Available: https://www.consumerfinance.gov/data-research/research-reports/chatbots-in-consumer-finance/chatbots-in-consumer-finance/

[151] Y. Li, S. Wang, H. Ding, and H. Chen, "Large language models in finance: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2311.10723

[152] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, "Large language models in medicine." 2023. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/349830

[153] D. Jin, E. Pan, N. Oufattole, W.-H. Weng, H. Fang, and P. Szolovits, "What disease does this patient have? a large-scale open domain question answering dataset from medical exams," 2020. [Online]. Available: https://arxiv.org/abs/2009.13081

[154] E. Horvitz, "The power of prompting," *Microsoft Research Blog*, 2023, accessed: 2025-07-16. [Online]. Available: https://www.microsoft.com/en-us/research/blog/the-power-of-prompting/

[155] M. B. II and D. M. Katz, "Gpt takes the bar exam," 2022. [Online]. Available: https://arxiv.org/abs/2212.14402

[156] J. Purtill, "How chatgpt and other new ai tools are being used by lawyers, architects and coders," *ABC News*, 2023, accessed: 2025-07-17. [Online]. Available: https://www.abc.net.au/news/science/2023-01-25/chatgpt-midjourney-generative-ai-and-future-of-work/101882580

[157] K. Y. Iu and V. M.-Y. Wong, "ChatGPT by OpenAI: The End of Litigation Lawyers?" Rochester, NY, Jan. 2023. [Online]. Available: https://papers.ssrn.com/abstract=4339839

[158] N. Xie, Y. Bai, H. Gao, Z. Xue, F. Fang, Q. Zhao, Z. Li, L. Zhu, S. Ni, and M. Yang, "Delilaw: A chinese legal counselling system based on a large language model," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, ser. CIKM '24. ACM, Oct. 2024, p. 5299–5303. [Online]. Available: http://dx.doi.org/10.1145/3627673.3679219

[159] J. Cui, M. Ning, Z. Li, B. Chen, Y. Yan, H. Li, B. Ling, Y. Tian, and L. Yuan, "Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model," 2024. [Online]. Available: https://arxiv.org/abs/2306.16092

[160] Z. Zhang, D. Zhang-Li, J. Yu, L. Gong, J. Zhou, Z. Hao, J. Jiang, J. Cao, H. Liu, Z. Liu, L. Hou, and J. Li, "Simulating classroom education with llm-empowered agents," 2024. [Online]. Available: https://arxiv.org/abs/2406.19226

[161] Y. Chen, N. Ding, H.-T. Zheng, Z. Liu, M. Sun, and B. Zhou, "Empowering private tutoring by chaining large language models," 2024. [Online]. Available: https://arxiv.org/abs/2309.08112

[162] A. Zentner, "Applied innovation: Artificial intelligence in higher education," *SSRN Electronic Journal*, 12 2022.

[163] B. Zhang, "Preparing educators and students for chatgpt and ai technology in higher education:benefits, limitations, strategies, and implications of chatgpt ai technologies." 01 2023.

[164] Y. K. Dwivedi, N. Kshetri, L. Hughes, E. L. Slade, A. Jeyaraj, A. K. Kar, A. M. Baabdullah, A. Koohang, and V. R. et al., "Opinion paper: "so what if chatgpt wrote it?" multidisciplinary perspectives on opportunities, challenges and implications of generative conversational ai for research, practice and policy," *International Journal of Information Management*, vol. 71, p. 102642, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0268401223000233

[165] Y. Chen, S. Jensen, L. Albert, S. Gupta, and T. Lee, "Artificial intelligence (ai) student assistants in the classroom: Designing chatbots to support student success," *Information Systems Frontiers*, vol. 25, 06 2022.

[166] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of large language models (llms): A survey," 2024. [Online]. Available: https://arxiv.org/abs/2403.05156

[167] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," 2021. [Online]. Available: https://arxiv.org/abs/2012.07805

[168] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," pp. 1322–1333, 10 2015.

[169] F. Leboukh, E. Aduku, and O. Ali, "Balancing chatgpt and data protection in germany: Challenges and opportunities for policy makers," *Journal of Politics and Ethics in New Technologies and AI*, vol. 2, p. e35166, 08 2023.

[170] J. Domingo-Ferrer, N. Jebreel, and D. Sánchez, "Efficient unlearning with privacy guarantees," 2025. [Online]. Available: https://arxiv.org/abs/2507.04771

[171] P. V. Falade, "Decoding the threat landscape: Chatgpt, fraudgpt, and wormgpt in social engineering attacks," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, p. 185–198, Oct. 2023. [Online]. Available: http://dx.doi.org/10.32628/CSEIT2390533

[172] T. V. Doan, Z. Chu, Z. Wang, and W. Zhang, "Fairness definitions in language models explained," 2024. [Online]. Available: https://arxiv.org/abs/2407.18454

[173] E. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" pp. 610–623, 03 2021.

[174] N. Meade, E. Poole-Dayan, and S. Reddy, "An empirical survey of the effectiveness of debiasing techniques for pre-trained language models," 2022. [Online]. Available: https://arxiv.org/abs/2110.08527

[175] N. Saxena, W. Zhang, and C. Shahabi, *Missed Opportunities in Fair AI*, 04 2023, pp. 961–964.

[176] I. O. Gallegos, R. A. Rossi, J. Barrow, M. M. Tanjim, S. Kim, F. Dernoncourt, T. Yu, R. Zhang, and N. K. Ahmed, "Bias and fairness in large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2309.00770

[177] A. Pal, L. K. Umapathi, and M. Sankarasubbu, "Med-halt: Medical domain hallucination test for large language models," 2023. [Online]. Available: https://arxiv.org/abs/2307.15343

[178] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022. [Online]. Available: https://arxiv.org/abs/2203.02155

[179] Z. Yu, Y. Wu, N. Zhang, C. Wang, Y. Vorobeychik, and C. Xiao, "Codeipprompt: Intellectual property infringement assessment of code language models," *Proceedings of Machine Learning Research*, vol. 202, pp. 40 373–40 389, 2023, publisher

[180] J. Dastin, "Microsoft attracting users to its code-writing, generative ai software," *Reuters*, 2023, accessed: 2025-07-19. [Online]. Available: https://www.reuters.com/technology/microsoft-attracting-users-its-code-writing-generative-ai-software-2023-01-25/

[181] D. Milmo, "Sarah silverman sues openai and meta claiming ai training infringed copyright," *The Guardian*, 2023, accessed: 2025-07-19. [Online]. Available: https://www.theguardian.com/technology/2023/jul/10/sarah-silverman-sues-openai-meta-copyright-infringement

[182] J. Stempel, "Ny times sues openai, microsoft for infringing copyrighted works," *Reuters*, 2023, accessed: 2025-07-19. [Online]. Available: https://www.reuters.com/legal/transactional/ny-times-sues-openai-microsoft-infringing-copyrighted-work-2023-12-27/

[183] L. Bakare, "An ai-generated band got 1m plays on spotify. now music insiders say listeners should be warned," *The Guardian*, 2025, accessed: 2025-07-19.

[184] V. Fauchoux, "Ai ip: what lessons can be learned from the global buzz surrounding the generation of videos in the style of studio openai ghibli," *Deprez Guignot Associés*, 2025, accessed: 2025-07-19.

[185] Z. Li, C. Wang, S. Wang, and C. Gao, "Protecting intellectual property of large language model-based code generation apis via watermarks," 11 2023, pp. 2336–2350.

[186] D. Alon and J. Ko, "Goemotions: A dataset for fine-grained emotion classification," 2021, accessed: 2025-07-30. [Online]. Available: https://research.google/blog/goemotions-a-dataset-for-fine-grained-emotion-classification/

[187] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.