universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Leveraging Large Language Models to Identify Disinformation on Mastodon

verfasst von | submitted by

Artur Khaialiev

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Informatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Math. Dr. Peter Reichl Privatdoz.

Acknowledgements

I would like to express my sincere gratitude to everyone who supported me throughout the completion of this master's thesis. First and foremost, I am deeply grateful to my wife Daiana for her endless patience, support and encouragement during this challenging journey. I would also like to thank my family, especially my mother, for their constant motivation and belief in me. I extend my sincere thanks to my supervisors, Dipl.-Ing. Paul Fuxjäger and Univ.-Prof. Dipl.-Math. Dr. Peter Reichl, for their valuable guidance, expertise, and constructive feedback throughout my research. Thank you all for making this possible.

This research supports transparency in AI systems, so it is important to mention that Large Language Models (Claude and ChatGPT) helped with writing and improving parts of this research. Since this thesis tests LLMs for content moderation on decentralized social network, it makes sense to be honest about using these same technologies for writing. The AI tools helped make the text clearer and better organized.

Abstract

Disinformation on social media is a growing problem, but detecting it on decentralized platforms like Mastodon is particularly challenging. Unlike centralized networks where one company controls moderation, Mastodon consists of thousands of independent servers, making it difficult to implement consistent detection systems. The goal of this master thesis is to explore how Large Language Models (LLMs) can identify disinformation on Mastodon, with a focus on comparing how different models perform on the same data.

This study implements a federated feed parser to collect posts from multiple Mastodon servers in real-time using the ActivityPub protocol. The detection approach combines content analysis with behavioral heuristics based on account characteristics such as account age, follower counts and posting patterns. These heuristics serve as a pre-filtering mechanism to reduce the volume of content requiring expensive LLM analysis, making the approach feasible for resource-constrained instances.

The main contribution of this work is a systematic comparative analysis of four state-of-the-art LLM architectures: OpenAI's GPT-4o-mini, Anthropic's Claude Sonnet 4, Google's Gemini 2.5 Flash Lite and Meta's Llama 3.3 70B, evaluated on identical Mastodon datasets collected from the public federated timeline. Each model analyzes the same flagged posts and provides confidence scores and classification decisions, allowing for direct comparison of their outputs. This multi-model approach reveals how different LLMs interpret the same content and identifies patterns in their agreement and disagreement.

Results show substantial inter-model agreement, with most posts receiving consistent classifications across all four models. The behavioral filtering approach significantly reduced the analysis workload, though without ground truth labels, this study measures inter-model consistency rather than detection accuracy.

Importantly, this system is designed as a decision-support tool to assist human moderators rather than replace them. All automated classifications serve as recommendations for human review, helping moderators prioritize their efforts while maintaining human oversight of all moderation actions.

This thesis provides practical guidance for Mastodon administrators seeking to implement automated content analysis, establishes benchmarks for comparing LLM-based detection systems in decentralized social networks and contributes to the broader understanding of how different AI architectures approach content moderation challenges. The comparative framework developed in this work can be adapted for other federated platforms and extended to evaluate future LLM models.

Kurzfassung

Desinformation in sozialen Medien ist ein wachsendes Problem, aber ihre Erkennung auf dezentralen Plattformen wie Mastodon ist besonders herausfordernd. Im Gegensatz zu zentralisierten Netzwerken, bei denen ein Unternehmen die Moderation kontrolliert, besteht Mastodon aus Tausenden unabhängiger Server, was die Implementierung konsistenter Erkennungssysteme erschwert. Ziel dieser Masterarbeit ist es zu untersuchen, wie Large Language Models (LLMs) Desinformation auf Mastodon identifizieren können, mit Schwerpunkt auf dem Vergleich der Leistung verschiedener Modelle bei denselben Daten.

Diese Studie implementiert einen föderierten Feed-Parser zur Erfassung von Beiträgen mehrerer Mastodon-Server in Echtzeit unter Verwendung des ActivityPub-Protokolls. Der Erkennungsansatz kombiniert Inhaltsanalyse mit verhaltensbasierten Heuristiken, die auf Kontomerkmalen wie Kontoalter, Follower-Anzahl und Posting-Mustern basieren. Diese Heuristiken dienen als Vorfilter-Mechanismus, um das Volumen der Inhalte zu reduzieren, die eine kostspielige LLM-Analyse erfordern, wodurch der Ansatz für ressourcenbeschränkte Instanzen praktikabel wird.

Der Hauptbeitrag dieser Arbeit ist eine systematische vergleichende Analyse von vier hochmodernen LLM-Architekturen: OpenAIs GPT-4o-mini, Anthropics Claude Sonnet 4, Googles Gemini 2.5 Flash Lite und Metas Llama 3.3 70B, die anhand identischer Mastodon-Datensätze aus der öffentlichen föderierten Timeline evaluiert wurden. Jedes Modell analysiert dieselben markierten Beiträge und liefert Konfidenzwerte sowie Klassifizierungsentscheidungen, was einen direkten Vergleich ihrer Ausgaben ermöglicht. Dieser Multi-Modell-Ansatz zeigt, wie verschiedene LLMs denselben Inhalt interpretieren, und identifiziert Muster in ihrer Übereinstimmung und Uneinigkeit.

Die Ergebnisse zeigen eine erhebliche Übereinstimmung zwischen den Modellen, wobei die meisten Beiträge konsistente Klassifizierungen über alle vier Modelle hinweg erhalten. Der verhaltensbasierte Filteransatz reduzierte die Analysearbeitslast erheblich, obwohl diese Studie ohne Ground-Truth-Labels die Modellkonsistenz statt der Erkennungsgenauigkeit misst.

Wichtig ist, dass dieses System als Entscheidungsunterstützungswerkzeug konzipiert ist, um menschliche Moderatoren zu unterstützen, nicht zu ersetzen. Alle automatisierten Klassifizierungen dienen als Empfehlungen zur menschlichen Überprüfung und helfen Moderatoren, ihre Bemühungen zu priorisieren, während die menschliche Aufsicht über alle Moderationsmaßnahmen erhalten bleibt.

Diese Arbeit bietet praktische Anleitungen für Mastodon-Administratoren, die automatisierte Inhaltsanalyse implementieren möchten, etabliert Benchmarks für den Vergleich LLM-basierter Erkennungssysteme in dezentralen sozialen Netzwerken und trägt zum breiteren Verständnis bei, wie verschiedene KI-Architekturen Herausforderungen der Inhaltsmoderation angehen. Das in dieser Arbeit entwickelte Vergleichsframework kann für

andere föderierte Plattformen angepasst und zur Evaluierung zukünftiger LLM-Modelle erweitert werden.

Contents

List of Tables

List of Figures

Listings

1. Introduction

Social media platforms are now important tools that people use every day to communicate. Billions of users share information and participate in public discussions on these platforms. However, social media has also become a place where disinformation spreads easily. Disinformation means false or misleading information that someone creates on purpose to trick people [ZZ20]. This problem can cause real harm, such as affecting election results or spreading incorrect health information during pandemics [LBB+18].

Major platforms like X (formerly Twitter) and Facebook spend significant resources fighting disinformation through human moderators, automated tools and fact-checking partnerships. Despite these efforts, the volume and complexity of disinformation campaigns continue to overwhelm even well-funded platforms [HV22]. The challenge becomes even greater for decentralized social networks, where no single organization controls content moderation.

Decentralized social networks aren't completely new. Diaspora, launched in 2011, was one of the first attempts to build a social platform without centralized control [BHG+12]. Its creators emphasized that "our distributed design means that no big corporation will ever control Diaspora" [GSSZ]. This approach prioritizes user autonomy, data privacy and community-based moderation.

Mastodon, created in 2016, is one of the most successful examples of this decentralized approach. As an open-source micro-blogging platform, Mastodon works through a federation of independent servers called *instances*. Each instance has its own administrators with their own moderation rules [RJDC+19]. Users can post and share content from others and interact across instances using the ActivityPub protocol. However, there's no central authority that can enforce uniform content moderation across the entire network.

After major events like Elon Musk's purchase of X in 2022, Mastodon saw significant user growth [RAL+25]. People moved to Mastodon due to concerns about content moderation and algorithm changes. Today, the platform has over 8800 active instances and 253000 monthly active users [Mas25].

Recently, another decentralized platform has appeared: Bluesky. The platform opened to the public in 2024 and quickly grew to over 30 million users [Wikb]. A major growth period happened during the November 2024 U.S. presidential election. At that time, Elon Musk publicly supported Donald Trump [All24], which caused another wave of users to leave X and join alternative platforms like Bluesky [Wikc].

While Bluesky has grown quickly, this research focuses on Mastodon for several important reasons. First, Mastodon has been around longer, giving us more data on user behavior. Second, being open-source makes it more transparent and accessible for research, with publicly available APIs and tools such as Mastodon.py enabling straightforward

data collection. Third, Mastodon's federated structure creates unique content moderation challenges that represent broader issues in decentralized systems.

1.1. Why Mastodon Needs Automated Disinformation Detection

The decentralized nature of Mastodon creates unique challenges for identifying disinformation. Instance administrators (many of whom are volunteers) must moderate content without access to the sophisticated tools available to large tech companies. While some instances have active moderation teams, others may lack the capacity to effectively review all content, particularly during high-activity periods or coordinated disinformation campaigns.

Traditional methods for checking content work well on centralized platforms, but they are hard to use in federated environments. On platforms like X or Facebook, one company controls everything. This makes it easier to use the same detection methods everywhere and respond quickly to new problems. Mastodon works differently because it uses a federated structure meaning each instance runs independently. An administrator who manages one instance cannot control content on other instances, even when their users can see that content.

Cross-instance coordination is limited, making it challenging to track how potential disinformation spreads across the network. A false claim might originate on one instance, be shared to another and eventually reach users across dozens of servers, each with different moderation capacities and priorities. The diverse moderation philosophies across instances mean that content considered problematic on one server might be acceptable on another. Some instances prioritize free speech, while others implement strict content policies.

The technical infrastructure of many Mastodon instances also presents challenges. Unlike major tech companies that can deploy machine learning models on powerful servers, many instance administrators operate on limited budgets with simple hardware. They need detection solutions that are efficient, easy to implement and compatible with existing workflows. Additionally, any automated system must be transparent and explainable, allowing moderators to understand why content was flagged.

Automated detection systems that use artificial intelligence could help solve these problems. Recent progress in Large Language Models (LLMs) has shown that these models are very good at understanding language and finding patterns [BMR+20, BHA+21]. Model families like OpenAI's GPT-4, Anthropic's Claude, Google's Gemini and Meta's Llama can analyze large amounts of text and find language patterns that might indicate for disinformation. Traditional systems use simple rules and search for specific keywords. LLMs are different because they can understand context, find hidden manipulation methods and adjust to new strategies that disinformation spreaders use.

However, their application to decentralized social networks remains largely unexplored. Existing datasets and research on LLM-based content moderation have primarily focused on centralized platforms such as Twitter and Reddit [KRDE25]. Questions remain

about how well these models perform on the shorter, conversational content typical of micro-blogging platforms and how they handle linguistic diversity across Mastodon's international user base.

An automated detection system could assist instance administrators by flagging potentially misleading content for review. Behavioral indicators can serve as efficient pre-filters, making comprehensive LLM analysis feasible even for resource-limited instances.

1.2. Research Problem and Motivation

Despite the growing need for disinformation detection on Mastodon, existing solutions are limited. Most research on automated detection has focused on centralized platforms like X, where researchers have access to comprehensive datasets and uniform platform features [FVD+16]. The unique characteristics of federated networks such as distributed content, varied moderation policies and cross-instance interactions require different approaches.

Academic literature on disinformation detection has largely developed around centralized platforms, where certain assumptions work that may not apply to decentralized networks. For example, many detection methods rely on network analysis techniques that assume complete visibility of user connections. In Mastodon's federated environment, such complete visibility is not guaranteed. Additionally, user behavior patterns may differ significantly between centralized and decentralized platforms.

Furthermore, while LLMs have shown promise in various content moderation tasks, their effectiveness for disinformation detection in decentralized environments hasn't been systematically tested. Different model architectures may have different strengths and weaknesses when analyzing social media content. Understanding these differences is important for making decisions about which approaches to use in resource-limited environments.

Recent developments in the LLM area have introduced several distinct approaches. OpenAI's GPT represents the latest generation of transformer-based models trained on massive datasets [AAA+23]. Anthropic's Claude uses Constitutional AI training methods designed to make the model more helpful and honest [BKK+22]. Google's Gemini brings multimodal capabilities and integration with Google's knowledge bases [Gem23b]. Meta's Llama, as an open-source alternative, provides transparency in model architecture [TMS+23].

Each of these models represents different approaches to AI development. However, their comparative performance on disinformation detection in federated social networks remains unexplored. Understanding how these models compare on identical datasets is important for several reasons. First, different models may be better at detecting different types of disinformation. Second, models trained with different methods may show different biases.

This research addresses these gaps by developing and evaluating an LLM-based framework for detecting disinformation on Mastodon. The study implements a real-time federated feed parser that monitors the public timeline using the ActivityPub protocol, capturing not only post content but also metadata about users and their behavior patterns.

The detection framework works on multiple analytical dimensions. Content-based

analysis examines the text of posts using advanced natural language processing techniques. Account-based analysis considers user characteristics like account age and posting frequency. Behavioral heuristics marks suspicious patterns like newly created accounts posting high volumes of content on trending topics. The system implements a filtering mechanism that flags posts for detailed LLM analysis based on these behavioral heuristics. When a post shows suspicious characteristics, it's queued for analysis by four LLMs. This multi-model approach enables direct comparison of how different AI architectures interpret identical content under identical conditions.

The main contribution of this work is a systematic comparative analysis of four state-of-the-art LLM architectures (GPT-4, Claude, Gemini and Llama 3.3), applied to disinformation detection on Mastodon. By evaluating these models on identical datasets with consistent protocols and standardized metrics, this research reveals how each model interprets potentially misleading content and identifies their respective strengths and limitations.

1.3. Research Questions and Objectives

This thesis investigates the following research questions:

1. How can Large Language Models be used to detect disinformation in Mastodon's federated network and what technical infrastructure is needed to analyze real-time streaming data?

2. Which behavioral and content-based metrics work as pre-filters to reduce LLM workload and enhance contextual input for disinformation detection?

3. How do distinct LLM architectures compare in their ability to detect disinformation when tested on identical Mastodon datasets?

To address these questions, this research pursues the following objectives:

- Implement a real-time federated feed parser using the Mastodon streaming API to continuously monitor the public timeline and capture posts with associated metadata

- Develop behavioral heuristics for identifying suspicious posts based on account characteristics and content patterns

- Design and implement a multi-model analysis pipeline that queries four distinct LLM architectures for each flagged post

- Build a real-time interface that streams analysis results from all models, enabling transparent comparison

1.4. Scope and Limitations

This research focuses on text-based disinformation detection on Mastodon. While the platform supports various content types including images and videos, this study primarily examines text posts and their associated metadata. This focus makes sense because text remains the primary way people share information on social platforms and text analysis is what current LLMs do best.

Data collection focuses on Mastodon's public federated timeline, which aggregates posts from instances that have enabled federation. This provides access to diverse content from multiple instances and communities, but does not capture private posts or content from instances with limited federation.

The comparative analysis evaluates four families of LLM architectures to represent different approaches in AI development. This selection cannot be exhaustive due to practical constraints, but these four families represent distinct approaches that are both accessible to researchers and potentially implementable by instance administrators. These four model families were selected because they represent distinct approaches to LLM development (proprietary vs. open-source, different training methodologies) and are accessible through well-documented APIs.

The research develops a proof-of-concept system that demonstrates feasibility and enables comparative evaluation, rather than a production-ready deployment. The implemented system serves as a research platform for evaluating approaches and establishing benchmarks. It is important to note that this system is designed as a helper tool for human moderators, not as a replacement for them. The automated detection helps people make better decisions, but humans should always have the final word in the moderation.

2. Background and Related Work

The challenge of detecting disinformation in decentralized social networks is at the intersection of several research areas: content moderation, federated networks architecture and artificial intelligence. This chapter provides background on these areas and examines existing research relevant to this thesis. First, we explore disinformation and current detection approaches. Second, we examine Mastodon's decentralized architecture and its moderation challenges. Third, we introduce Large Language Models and their applications in content analysis. Finally, we review related research and identify gaps that this thesis addresses.

2.1. Disinformation in Social Media

2.1.1. Definitions and Taxonomy

The terms *disinformation*, *misinformation* and *malinformation* are often used to mean the same thing, but researchers make clear differences between them [TJLL18]. **Misinformation** means false information that someone shares without wanting to harm others: the person who shares it thinks it's true. **Disinformation** is different because it means someone creates and spreads false information on purpose to trick people or cause harm. **Malinformation** means real information that someone shares with bad intentions, like leaking private information to damage someone's reputation.

This research focuses primarily on disinformation, as it represents the most intentional and potentially harmful category. Disinformation campaigns can be highly sophisticated, using psychological manipulation, coordinated networks of accounts and strategic timing to maximize impact.

Disinformation can be divided into different types based on what it tries to achieve:

- **Political disinformation** tries to influence elections, damage democratic institutions or divide communities. Examples include false information about political candidates and fake claims about voter fraud. Large disinformation campaigns happened during the 2016 U.S. presidential election[AG17].
- **Health disinformation** spreads false medical information. During the COVID-19 pandemic, there were many false claims about vaccines and treatments [Iea20].
- **Commercial disinformation** promotes products through false claims or attacks competitors with fabricated allegations. Fake reviews manipulate buying decisions and cost businesses billions of dollars globally [TDKM23].

- **Social disinformation** targets specific groups or individuals, often inciting hatred or discrimination. This type of disinformation can lead to offline violence and causes serious harm to targeted communities [UBC22].

2.1.2. The Spread of Disinformation on Social Media

Social media platforms have completely changed how information spreads. Traditional media works through gatekeepers like editors and fact-checkers who check information before it gets published. Social media is different because it allows anyone to create and share content. Users can instantly reach very large audiences without any filters. This has many positive effects, but it also means that disinformation can spread very quickly without the quality controls that traditional media has.

Studies have shown why disinformation spreads so easily on social media. A major study by Vosoughi and colleagues analyzed how information moves through X. They found that false news travels much faster and reaches more people than true news. According to their research, false stories had a higher chance of being shared compared to accurate information [VRA18].

Confirmation bias means that people easily believe information that matches what they already think. At the same time, they carefully check or reject information that goes against their beliefs. This bias is very strong. Sometimes when people see facts that prove them wrong, they believe the false information even more strongly [Cas10]. Emotional content spreads faster than neutral information. Posts that make people angry or scared spread especially quickly [KGH14].

Network effects mean that when popular or influential users share disinformation, it reaches many more people than when regular users share the same content. Social media works very fast, which makes this problem worse. False information can reach millions of people before fact-checkers can verify it and respond. Moreover, when corrections are published later, they often arrive too late to have significant effect, as most engagement with the original false post has already occurred [SUS25].

Coordinated disinformation campaigns use these effects in a planned and organized way. Groups of accounts work together to spread specific messages and make certain topics look popular or trending. These accounts can be automated bots or they can be controlled by real people [FVD+16].

2.1.3. Challenges in Detection

Disinformation has effects that go beyond single false posts. Political disinformation damages democratic processes[AG17]. Health disinformation during the COVID-19 pandemic made people doubt vaccines and try dangerous treatments on their own [Zar20].

Detecting and fighting disinformation is difficult for many reasons. Automated systems often have problems making these detailed judgments. Cultural and language differences affect how people understand information which makes it hard to create detection rules that work everywhere: what counts as disinformation can be different in different cultures

and societies. Language features like sarcasm or irony are especially hard for automated systems to understand correctly.

People who spread disinformation constantly change their methods to avoid detection. This creates a continuous arms race where detection methods become less effective once the spreaders learn about them and develop more advanced evasive techniques [MH25].

2.2. Mastodon as a Decentralized Network

2.2.1. Architecture and Technical Foundation

Mastodon represents a fundamentally different approach to social networking compared to centralized platforms. Rather than operating as a single service controlled by one organization, Mastodon functions as a federation of independent servers [ZGR18].

This architecture is built on the ActivityPub protocol, a W3C standard for decentralized social networking [act18]. ActivityPub enables instances to communicate with each other, allowing users on different servers to follow each other and interact as if they were on the same platform. Each Mastodon instance maintains its own database of local users and content, plus cached copies of remote content relevant to its users. When federation occurs, instances exchange messages using ActivityPub's standardized format.

The federation model has several advantages. Users can choose which instance to join based on their preferences for community rules and moderation style. Instance administrators can create rules that fit their communities without needing agreement from the entire network. The distributed structure also makes the system more stable: the network keeps working even if some instances go offline due to the decentralized nature.

However, this structure also creates problems. On centralized platforms, one company stores all data in a centralized database. In federated networks information is spread across thousands of independent servers, which makes many things more complicated.

2.2.2. Growth and User Migration

Mastodon has grown significantly, especially when users became unhappy with centralized platforms. After Elon Musk bought X in 2022, many users moved to Mastodon [RAL+25]. The platform now has over 8800 active instances and about 253000 monthly active users [Mas25].

People join Mastodon for different reasons: some value privacy and community independence, other want to avoid corporate control. Research shows that Mastodon users care more about privacy than typical social media users [RJDC+19].

2.2.3. Content Moderation Challenges

Content moderation on Mastodon is different from centralized platforms. On X or Facebook, company hires moderators who enforce the same rules across the entire platform using centralized tools. On Mastodon moderation is spread across many instances. Instance administrators face several problems. Most of them are volunteers who have

limited time and resources [RJDC+19]. They need to moderate content from both their own users and content that comes from other instances through federation. Administrators can control content on their own instance, but they cannot control what happens on other instances. The amount of moderation work can be very large. Even small instances create a lot of content that potentially needs to be checked. Without the resources that big platforms have, many administrators struggle with this workload.

Mastodon provides several tools for managing federation. Administrators can block entire instances or limit them. Individual users can also block accounts or mute domains. These tools help communities maintain their culture, but they also mean that different parts of the network see very different content which creates problems when dealing with coordinated disinformation campaigns. A campaign might start on one instance, spread through federation to other instances and each instance's moderators might handle it differently.

2.3. Large Language Models for Content Analysis

2.3.1. Transformer Architecture and Foundation

Large Language Models represent a major advance in Natural Language Processing. They build on the transformer architecture that Vaswani et al. introduced [Vea17]. Earlier approaches processed text word by word in order. Transformers work differently: they use attention mechanisms to look at relationships between all words in a text at the same time.

The transformer architecture has encoder and decoder components, as shown in Figure 2.1. The key innovation is the self-attention mechanism. This allows the model to decide how important different words are when it processes each word in a sequence.

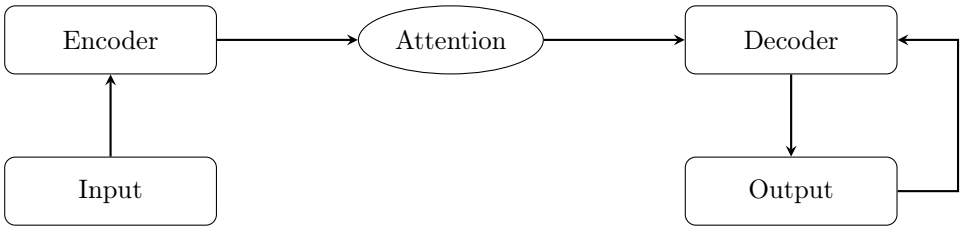

Figure 2.1.: Simplified Encoder-Decoder architecture of transformer models

The word **large** in Large Language Models refers to two things: the number of parameters and the amount of training data. These models are trained on huge amounts of text usually scrapped from internet. During training, they learn statistical patterns that let them perform many different tasks without being trained specifically for each task [BMR+20].

2.3.2. GPT-4

GPT (Generative Pre-trained Transformer) models are developed by OpenAI. They use a decoder-only transformer architecture designed for generating text. GPT-4, released in 2023, demonstrated significant improvements in reasoning capabilities and instruction following [AAA+23]. For detecting disinformation, GPT-4 has several advantages: its language understanding helps identify logical inconsistencies or emotional manipulation in text. However, like other LLMs, the model can generate confident but incorrect outputs, a phenomenon known as hallucination [FKKG24].

2.3.3. Claude

Claude is developed by Anthropic and focuses on safety, accuracy and alignment with human values [BKK+22]. Claude uses a training method called Constitutional AI that is designed to make the model "more helpful and honest". The Constitutional AI approach tries to create models that are more careful when they are not certain about something. For content moderation these features might be especially important because mistakes can seriously impact how people form opinions and understand information.

2.3.4. Gemini

Gemini is developed by Google DeepMind and features a natively multimodal architecture, meaning it can process different types of input including text, images, audio, and video simultaneously [Gem23a]. The Gemini family consists of three model sizes (Ultra, Pro, and Nano) suitable for different deployment scenarios. For disinformation detection, Gemini's cross-modal reasoning capabilities could be valuable for analyzing multimedia content and its integration with Google's knowledge infrastructure could potentially enhance fact-checking tasks.

2.3.5. Llama 3.3

Llama is developed by Meta and released as open-source, meaning the model architecture and training details are publicly available [TMS+23]. Researchers can examine how the model works and potentially fine-tune it for specific tasks.

The open-source nature creates several possibilities that proprietary models cannot offer. Researchers can customize the model using data from specific domains, as demonstrated by studies adapting Llama for clinical and other specialized applications [GDS+24]. The models from this family can be modified to better serve particular communities. Transparency also allows independent researchers to audit how the model behaves.

2.3.6. LLM-Based Classification Approaches

Applying LLMs to content moderation involves several approaches. **Zero-shot classification** uses carefully crafted prompts to ask models to classify content without task-specific training. Zero-shot classification is used in this research. **Few-shot learning** provides

a few examples to help models understand the task. **Fine-tuning** adapts pre-trained models specifically for disinformation detection using labeled datasets.

Research has demonstrated that LLMs can identify various forms of problematic content, including hate speech and toxic comments [RHMS23]. However, most studies focus on centralized platforms. The application to decentralized networks like Mastodon remains underexplored.

2.4. Related Research and Identified Gaps

2.4.1. Behavioral Detection and Decentralized Platforms

Beyond analyzing content, researchers have studied behavioral indicators to identify suspicious accounts. Bot detection research examines features like posting frequency and timing [FVD+16], while coordinated behavior detection identifies multiple accounts performing synchronized actions [YVD+19]. However, research on decentralized social networks remains limited. Studies of Mastodon have primarily examined network structure and community dynamics rather than content moderation [ZGR18, LCGT21]. Raman et al. [RJDC+19] examined challenges facing volunteer moderators, highlighting resource constraints and lack of sophisticated tools, though their work focused on practices rather than technical solutions.

2.4.2. LLM-Based Content Moderation

Recent research has started to explore how Large Language Models can be used for content moderation tasks. Studies have shown that LLMs can perform well when detecting problematic content. Li et al. [LFAH24] found that ChatGPT achieved approximately 80% accuracy in classifying hateful, offensive and toxic content when compared to human annotations. However, most early studies tested only one model at a time. Kumar et al. [KBD24] addressed this gap by comparing multiple LLMs on toxicity detection tasks. Their results showed that LLMs significantly outperform traditional toxicity classifiers like the Perspective API. Interestingly, larger models did not always perform better, suggesting potential limits to scaling benefits for this task. These findings indicate that LLMs hold promise for content moderation, but systematic comparisons across architectures and platforms remain limited.

2.4.3. Comparison with Machine Learning Approaches

Before Large Language Models became popular, researchers used traditional machine learning methods for disinformation detection. These methods include Support Vector Machines (SVM), Random Forests, Decision Trees and Naive Bayes classifiers [ATS17]. These approaches require manual feature engineering: researchers must decide which characteristics of posts to measure and how to represent them numerically. Traditional ML approaches have several advantages. They are computationally cheap compared to LLMs. A trained SVM can classify thousands of posts per second on basic hardware.

They are also interpretable, because researchers can see exactly which features led to each decision. For instance, if a Random Forest flags a post, we can see that it was because of high hashtag count combined with low account age. However, traditional approaches have significant limitations for our use case:

1. **Feature engineering requirements**: someone must manually define what features matter. This requires domain expertise and constant updates as disinformation tactics change.
2. **Limited context understanding**: traditional models treat text as a bag of words or simple n-grams. They cannot understand meaning or detect sarcasm.
3. **Training data requirements**: these models need large labeled datasets for training. Creating such datasets for Mastodon specifically would require significant manual effort.
4. **Language limitations**: models trained on English data could perform poorly on other languages, so separate models would be needed for each language.

LLMs address most of these limitations. They do not require feature engineering because they learn representations automatically. They understand context and can detect more interesting patterns. They also work across languages because they were trained on multilingual data. The main disadvantage is computational cost, which our behavioral filtering approach helps address.

2.4.4. Identified Research Gaps

The literature review shows several important gaps:

1. **Focus on decentralized platforms**: researchers have studied disinformation detection a lot on centralized platforms. However, decentralized networks have not been studied much.
2. **Systematic comparison of multiple models**: studies that compare multiple LLM architectures using the same methods are rare.
3. **Combining behavioral and content signals**: most research treats disinformation detection as only a content classification problem. Research pays less attention to behavioral indicators.

This thesis addresses these gaps by developing and testing LLM-based disinformation detection methods specifically for Mastodon's federated network. It conducts systematic comparisons of four different model architectures and combines behavioral and content-based signals. The next chapter describes the methodology used to achieve these objectives.

3. Methodology

This chapter describes the methodology used to develop and test an automated disinformation detection system for Mastodon's federated network. The research implements a framework that combines real-time data collection, behavioral heuristics for filtering suspicious content and multi-model Large Language Model analysis. The following sections explain the system architecture, data collection procedures, content filtering mechanisms, LLM integration approach and evaluation framework in all details.

3.1. System Architecture Overview

3.1.1. Overall Design

The implemented system follows a modular architecture designed to handle real-time streaming data from Mastodon's federated timeline, apply behavioral heuristics to identify potentially suspicious content and analyze flagged posts using four distinct Large Language Models.

Figure 3.1 illustrates the system architecture. The system consists of four primary components:

1. **Data collection layer**: continuously monitors Mastodon's public federated timeline using the ActivityPub protocol, capturing posts and metadata in real-time.

2. **Filtering and storage layer**: applies behavioral heuristics to mark suspicious posts based on account characteristics and content patterns, storing both raw data and flagged content in a PostgreSQL database.

3. **LLM analysis layer**: processes flagged posts through four different Large Language Models (GPT-4o-mini, Claude Sonnet 4, Gemini 2.5 Flash Lite and Metas Llama 3.3 70B), collecting classification decisions, confidence scores and reasoning.

4. **Presentation layer**: provides a web-based interface for viewing flagged content, monitoring instances, tracking trends and comparing LLM outputs in real-time through WebSocket streaming.

The architecture is asynchronous and non-blocking, allowing continuous data collection to proceed independently of analysis operations. This design ensures the system can handle high-volume data streams without losing events, while computationally expensive LLM analysis occurs on-demand.

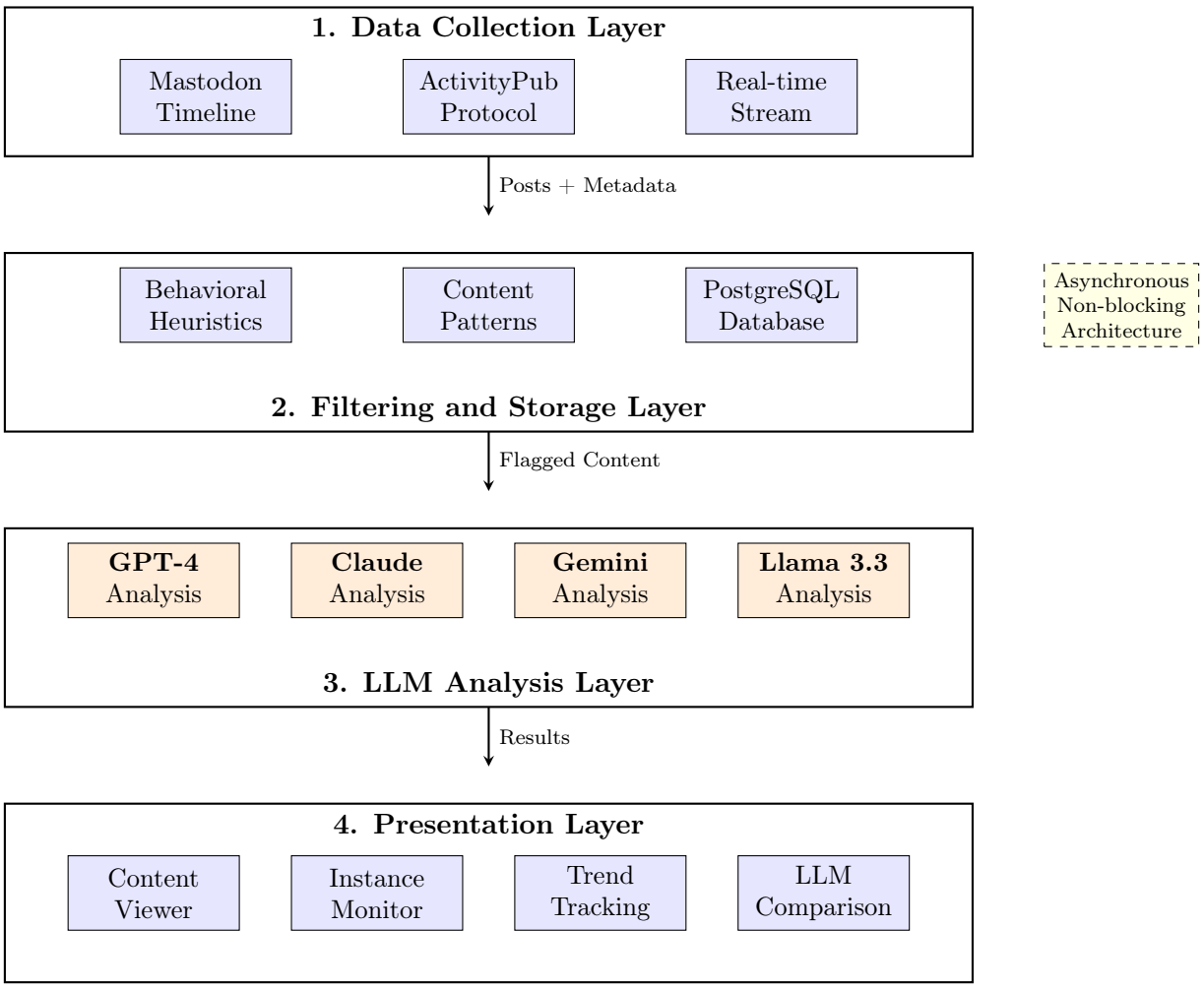

Figure 3.1.: High-level system architecture.

3.1.2. Technology Stack

The system leverages modern technologies selected for their performance for handling streaming data. Figure 3.2 summarizes the core technologies.

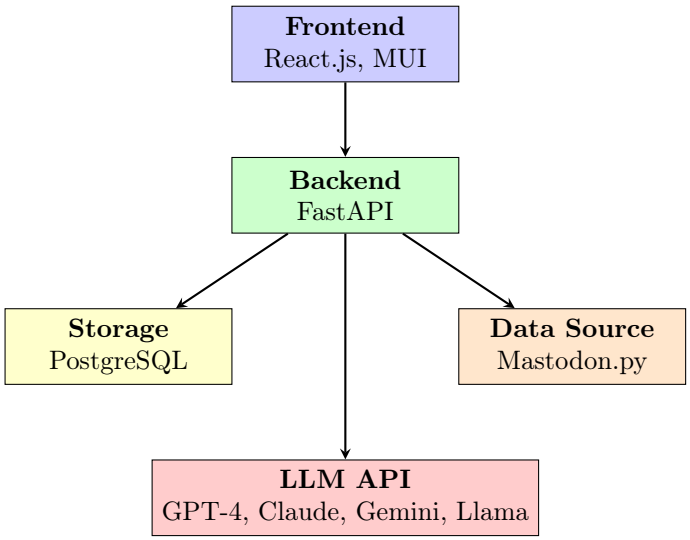

Figure 3.2.: High-level technology stack overview

Backend technologies: the backend uses FastAPI, a modern Python web framework chosen for asynchronous operations and high performance. PostgreSQL serves as the primary database, selected for its robust JSON support and powerful indexing capabilities (using GIN index). The Mastodon.py library provides Python bindings for the Mastodon API, including real-time streaming support.

Frontend technologies: the user interface uses React.js, a component-based JavaScript library that efficiently handles dynamic updates. Material-UI provides pre-built UI components. The frontend maintains WebSocket connections to receive real-time streaming responses from LLM models via FastAPI backend.

LLM integration: each LLM provider is accessed through its official SDK or API client: OpenAI's AsyncOpenAI client handles GPT-4 interactions, Anthropic's SDK provides streaming responses from Claude models, Google's Generative AI SDK integrates Gemini models and Llama 3.3 is accessed via Together AI's platform using an OpenAI-compatible API.

3.1.3. Data Flow Pipeline

The data flow through the system follows a carefully orchestrated sequence. Figure 3.3 illustrates the complete pipeline.

Stage 1: Real-Time Data Collection

1. The system establishes a persistent connection to Mastodon's public federated timeline.

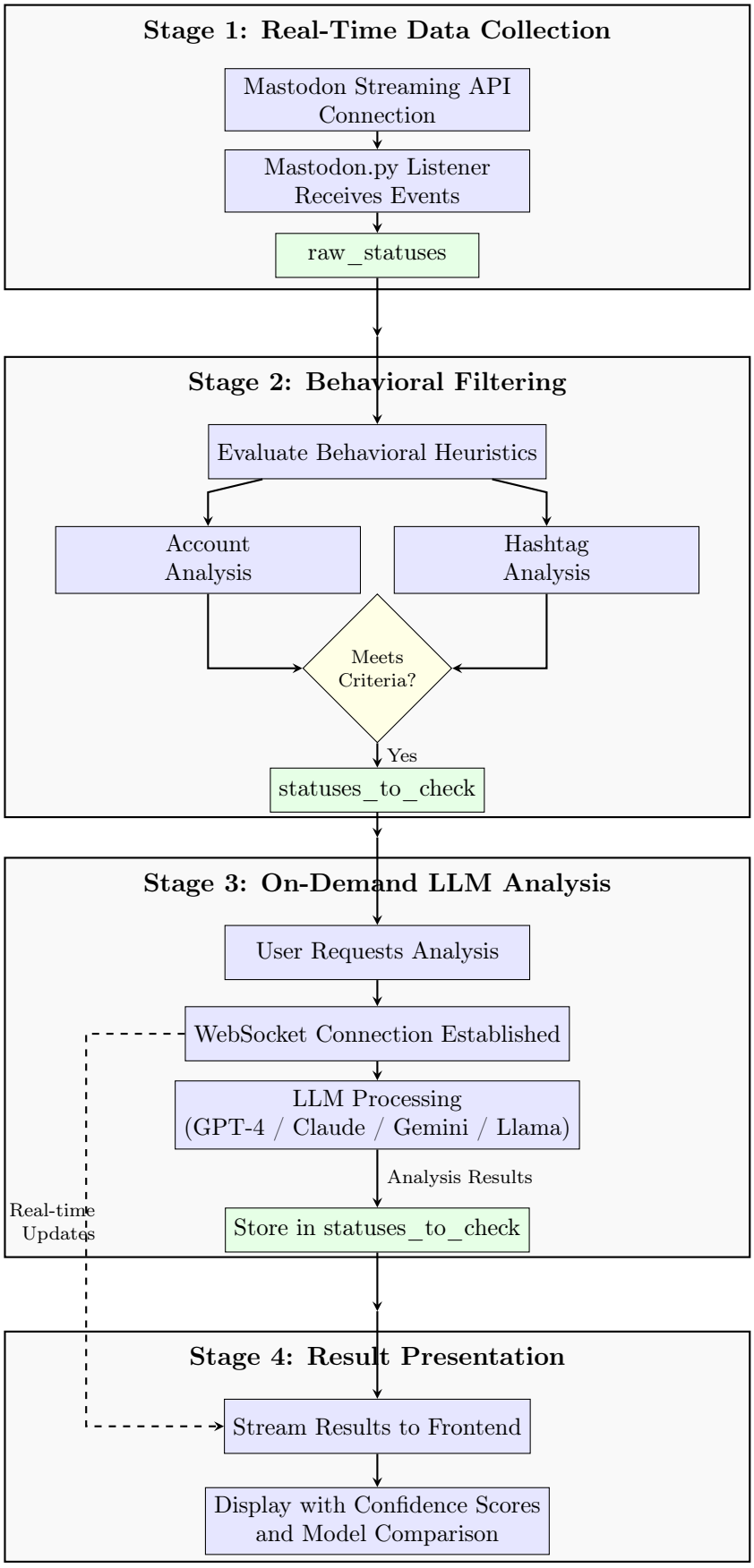

Figure 3.3.: Complete data flow pipeline.

2. As new posts appear, the Mastodon.py listener receives events in real-time.

3. Each event contains comprehensive post metadata including content, author information and other user metrics.

4. All received posts are stored in the `raw_statuses` table.

Stage 2: Behavioral Filtering

1. Posts are evaluated against behavioral heuristics to identify potentially suspicious content.

2. Account characteristics are extracted and compared against defined threshold values.

3. Posts using hashtags are analyzed to determine if they represent suspicious trending patterns.

4. Content similarity analysis compares new posts with previously flagged content.

5. Posts meeting suspicious criteria are stored in the `statuses_to_check` table.

Stage 3: On-Demand LLM Analysis

1. When a user requests analysis by sending an HTTP request, the system retrieves the post from the database.

2. A WebSocket connection is established between frontend and backend.

3. The post is submitted to the selected LLM.

4. Analysis results stream back in real-time.

5. Final responses are parsed and stored in model-specific columns.

Stage 4: Result Presentation

1. The frontend receives streaming updates via WebSocket.

2. When analysis completes, extracted metadata is displayed.

3. Users can compare analyses from different models side-by-side.

This pipeline architecture enables several important capabilities: continuous monitoring without losing data, efficient filtering to reduce costs of analysis and storage of all results for further research.

3.2. Data Collection and Preprocessing

3.2.1. Mastodon Streaming API Integration

The data collection component establishes a persistent connection to Mastodon's public federated timeline using the streaming API. Unlike traditional REST APIs that require periodic polling, the streaming API maintains an open connection and pushes new events to the client in real-time. The implementation uses the Mastodon.py library's `StreamListener` class, which provides an event-driven interface for handling incoming posts. Listing 3.1 shows the core structure.

```
class Listener(mastodon.StreamListener):
    def on_update(self, status):
        # Remove unnecessary nested objects
        status.pop("reblog", None)
        status.pop("media_attachments", None)

        # Store complete raw status
        with ScopedSession() as session:
            status_copy = copy(status)
            status_copy.pop("account", None)
            status_copy["tags"] = [item["name"]
                                   for item in status_copy["tags"]]
            status_copy["content"] = strip_html(
                status_copy["content"]
            )
            save_raw_status(session=session, status=status_copy)

        # Apply behavioral filtering
        if len(status["tags"]) != 0:
            apply_behavioral_heuristics(status)

async def listen_mastodon_stream():
    mastodon_instance.stream_public(
        Listener(),
        run_async=True
    )
```

Listing 3.1: Mastodon Stream Listener Implementation

The listener runs continuously as a background task. The streaming API provides access to the public federated timeline, aggregating posts from all instances that have enabled federation. Connection stability is critical for continuous data collection. The Mastodon.py library handles automatic reconnection in case of network interruptions.

3.2.2. ActivityPub Protocol Implementation

ActivityPub is a W3C standard protocol for decentralized social networking that enables communication between different instances [act18]. When a user creates a post, their home instance broadcasts an activity message to all instances where the user has followers

and federation is enabled. The system receives ActivityPub events in JSON format, containing full metadata about each post. Key fields extracted include:

- **Post content**: the text content originally in HTML format
- **Temporal information**: creation and edit timestamps of the posts
- **Engagement metrics**: counts of replies and favorites
- **Author information**: account details including registration date and follower counts
- **Hashtags**: array of hashtags used in the post
- **Language**: detected language code

This rich metadata enables multi-dimensional analysis combining content, behavior and user characteristics.

3.2.3. Data Preprocessing and Storage

Raw data is preprocessed before storage to ensure consistency and prepare content for analysis using the following steps:

HTML stripping: post content arrives in HTML format with markup tags. The `strip_html()` function removes all HTML tags while preserving text content, converting posts into plain text.

Object simplification: ActivityPub events contain deeply nested objects not needed for this research. Media attachments and emoji definitions are removed.

Hashtag normalization: hashtags are extracted into a simple array stored using PostgreSQL's native array type.

Timestamp standardization: all timestamps are converted to UTC and stored as PostgreSQL `DATETIME` fields.

The system maintains separate storage tables for different purposes, as shown in Table 3.1. The complete database schema can be found in Appendix B.

Table	Purpose
`raw_statuses`	All collected posts
`statuses_to_check`	Posts selected by pre-filters for LLM analysis, with analysis results
`accounts`	Author information for flagged posts
`suspicious_trends`	Hashtags identified as potentially artificial
`statuses`	Posts using flagged hashtags for similarity analysis

Table 3.1.: Database schema overview

The `raw_statuses` table serves as a comprehensive archive, enabling retrospective analysis. The `statuses_to_check` table contains only posts flagged as potentially suspicious, dramatically reducing the volume requiring LLM analysis.

3.3. Behavioral Heuristics for Content Filtering

A critical problem when using Large Language Models for disinformation detection is computational cost. Analyzing every post with multiple LLM models would be very expensive. To solve this problem, the system uses multi-stage filtering based on behavioral heuristics to identify posts requiring detailed analysis.

Initial filter: only posts containing at least one hashtag are evaluated by behavioral heuristics. Posts without hashtags are stored in the raw archive but not processed further. This decision reflects the research focus on detecting coordinated campaigns, which typically use hashtags to increase visibility and create artificial trends.

For posts with hashtags, the filtering mechanism combines three approaches: account-based indicators that check author metrics, trend analysis that identifies potentially artificial hashtag campaigns and content similarity analysis that detects coordinated posting patterns.

3.3.1. Account-Based Indicators

The first stage evaluates characteristics of the post author. The system applies three primary account-based criteria:

1. **Account age**: the system flags posts from accounts created within the last 30 days. Newly created accounts are often involved in coordinated campaigns because:
   - Campaign organizers can create multiple accounts at the same time
   - New accounts avoid reputational consequences of spreading disinformation
   - Moderation systems have less historical data for new accounts
   - Account creation is usually free on most instances
2. **Follower count**: posts from accounts with 1000 or fewer followers are flagged. This reflects that coordinated campaigns often have low organic engagement.
3. **Posts count**: accounts with 100 or fewer total posts are flagged. This identifies accounts with low posting history.

The combination of these three criteria creates a composite filter. An account must satisfy all three conditions for its posts to be flagged. Listing 3.2 demonstrates the implementation.

```
# Extract author information
account = status["account"]
created_at = account["created_at"].strftime("%Y-%m-%d %H:%M:%S")

# Calculate 30-day threshold
difference = (datetime.utcnow() - timedelta(days=30)).strftime("%Y-%m-%d
    %H:%M:%S")

# Apply composite filter
```

```
if (created_at >= difference
    and account["followers_count"] <= 1000
    and account["statuses_count"] <= 100):

    # Post meets suspicious account criteria
    flag_for_detailed_analysis(status, account)
```

Listing 3.2: Account-Based Filtering Implementation

These threshold values were chosen based on patterns identified in bot detection research rather than empirical optimization on our specific dataset. Account metadata features such as account age, follower count and posting frequency have been established as discriminative indicators for identifying automated accounts [VFD+17]. The 30-day account age threshold reflects common patterns observed in spam campaigns, where newly created accounts are frequently used for coordinated inauthentic activity. The follower and post count thresholds aim to identify accounts with limited organic engagement. However, these specific values were not validated through sensitivity analysis on Mastodon data and different thresholds might output different filtering results. Future work could systematically evaluate how varying these parameters affects both the volume of filtered content and the quality of subsequent LLM classifications.

3.3.2. Trend Analysis

Beyond individual account characteristics, the system analyzes hashtag usage patterns to mark potentially artificial trends. Coordinated campaigns often try to make specific hashtags trend by having multiple accounts post using the same tags at the same time.

The analysis operates in three stages. First, the system checks whether each hashtag represents a currently popular trend by querying the `trends` table, which is updated periodically with trending hashtags from Mastodon's API. Hashtags appearing in this list are excluded from suspicious trend analysis. Second, for hashtags not marked as popular trends, the system queries the Mastodon API to retrieve usage statistics over the previous seven days, including the number of unique accounts and total posts. Third, a hashtag is classified as potentially artificial if it meets both criteria: used by 10 or fewer unique accounts and appeared in 10 or fewer total posts. These thresholds identify hashtags in early stages of coordinated campaigns. When a hashtag meets these criteria, the system creates a record in the `suspicious_trends` table. Listing 3.3 shows the implementation.

```
for tag in status["tags"]:
    # Check if legitimate popular trend
    popular_trend = check_if_trend_popular(
        session=session,
        name=tag["name"]
    )

    # Skip popular trends
    if not popular_trend:
        # Check if already flagged as suspicious
        suspicious_trend = check_if_suspicious_trend_exist(
            session=session,
```

```
            name=tag["name"]
        )

        # Retrieve trend statistics if not cached
        if not suspicious_trend:
            url, accounts, uses, errors = (
                client.get_tag_info(tag=tag["name"])
            )

            # Flag as suspicious if low adoption
            if accounts <= 10 and uses <= 10:
                suspicious_trend = (
                    create_or_update_suspicious_trend(
                        session=session,
                        name=tag["name"],
                        url=url,
                        uses_in_last_seven_days=uses,
                        number_of_accounts=accounts,
                        instance_url=instance_url
                    )
                )
```

Listing 3.3: Trend Analysis Implementation

It is important to understand how Mastodon decides which hashtags are trending. The Mastodon's trending system works based on simple statistics: a hashtag becomes *trending* when many posts use it and many different accounts use it in a short period of time. There are no human moderators who check if trending content is true or false: hashtags become trending only because they are popular. This means that *legitimate trend* here means **popular trend** but **not verified**. If a disinformation campaign is large enough and well organized, it could make its hashtags trend. When this system asks mastodon.social's trends API for information, it gets that server's view of popular hashtags. The system does not check trending hashtags for suspicious content. This decision helps to save processing resources. However, this creates a known problem: if a disinformation campaign becomes popular enough to trend, it would not be detected by this part of the system, meaning the system may skip truly suspicious content

3.3.3. Content Similarity Analysis

The third component examines content similarity to detect coordinated campaigns where multiple accounts post identical or near-identical text.

Similarity computation: when a post passes the account-based filter (new account, low followers, few posts) and uses a hashtag flagged as potentially artificial, the system performs content similarity analysis before adding it to the LLM analysis queue. The comparison uses cosine similarity, which measures the angle between two document vectors represented through TF-IDF vectorization. Scores range from 0 (completely dissimilar) to 1 (identical).

Similarity threshold: cosine similarity score of 0.5 or higher indicates potentially coordinated content, increasing the `number_of_similar_statuses` counter. This threshold

is a practical choice: a similarity of 0.5 indicates substantial content overlap while still allowing for natural variation in phrasing. The threshold is intentionally moderate: lower values would flag posts that only share common words or general topics, while higher values would only catch posts that are almost identical. Listing 3.4 demonstrates the implementation.

```
from utils.similarity_checker import (
    calculate_cosine_similarity_between_two_statuses
)

# Retrieve all posts using this hashtag
statuses = get_statuses_by_tag(
    session=session,
    tag=tag["name"]
)

# Compare with each stored status
for stored_status in statuses:
    similarity = (
        calculate_cosine_similarity_between_two_statuses(
            status_content_1=stored_status.content,
            status_content_2=status["content"]
        )
    )

    # Flag coordinated content if similar
    if similarity >= 0.5:
        increment_suspicious_trend_number_of_similar_posts(
            session=session,
            suspicious_trend_id=suspicious_trend.id
        )
```

Listing 3.4: Content Similarity Analysis

3.3.4. Filtering and Resource Management

The federated timeline aggregates content from all instances connected through the ActivityPub protocol. For a well-connected instance like mastodon.social, the federated timeline typically receives between 1000 and 5000 posts per hour depending on time of day and network activity. This volume makes it infeasible to perform detailed LLM analysis on every post within the resource constraints of this research. The behavioral heuristics described above serve to reduce this volume to a manageable subset requiring detailed analysis. Through empirical observation during the evaluation period, the filtering system reduced the analysis workload to approximately 100-150 posts per hour. **It is important to note that high filtering efficiency does not mean high detection accuracy**: a system could achieve 99% filtering efficiency by simply selecting random posts, but this would not help the research goals. This behavioral heuristics aims to filter while keeping focus on posts that need closer examination using LLM models. This achieves several practical goals: it makes the work computationally feasible with limited research

resources and focuses on posts that show pre-defined suspicious behavioral signals. The filtering rate can be adjusted based on available resources and instance size.

3.3.5. Limitations and Vulnerabilities of Behavioral Heuristics

While behavioral heuristics used in this system is a practical way to reduce costs and make the research project feasible, these heuristics have important limitations that need critical examination.

Behavioral heuristics can be bypassed by people who understand the filtering criteria. Account aging is one vulnerability: one can create accounts long before they use them, which bypasses account age-based thresholds. Follower manipulation is another problem, where coordinated networks artificially increase follower counts through mutual following or by buying followers on special marketplaces. Also campaigns can avoid hashtag-based detection by not using hashtags at all. These vulnerabilities show a fundamental trade-off between computational efficiency and detection strength, because sophisticated campaigns could easily avoid detection completely by acting like legitimate accounts. The described behavioral heuristics represent one approach to the pre-filtering problem, chosen because they are simple and transparent. Their limitations provide valuable insights into the challenges of automated content moderation when dealing with evasion attempts.

Fixing these security gaps would need more research. Machine learning could spot complex patterns that basic rules miss. Network analysis could find coordinated attacks by looking at connections between accounts. Sharing data between servers could catch campaigns that spread across multiple platforms. These improvements would be useful but are beyond the scope of this proof-of-concept implementation.

3.4. Large Language Model Integration

3.4.1. Model Selection and Costs

This study tests four different LLM models that represent various ways of building AI systems. The chosen models include both major commercial options and open-source alternative.

GPT-4o-mini (by OpenAI): the system uses GPT-4o-mini, a variant maintaining strong performance while significantly reducing costs [AAA+23]. At current pricing, GPT-4o-mini costs $0.15 per million input tokens and $0.6 per million output tokens, with estimated cost per analysis of $0.0008-$0.0013.

Claude Sonnet 4 (by Anthropic): Claude Sonnet 4 uses Constitutional AI - a method for training AI systems to be "helpful, harmless, and honest" [BKK+22]. Claude costs $3 per million input tokens and $15 per million output tokens, with estimated cost per analysis of $0.020-$0.030.

Gemini 2.5 Flash Lite (by Google): Google's Gemini 2.5 Flash Lite provides a lightweight, cost-effective option designed for fast processing [Gem23a]. Gemini is priced at $0.1 per million input tokens and $0.4 per million output tokens, with estimated cost per analysis of $0.00055-$0.00085.

Llama 3.3 70B Instruct Turbo (by Meta): Meta's Llama 3.3 represents the open-source alternative, providing transparency [TMS+23]. Through Together AI, Llama costs $0.13 per million input tokens and $0.13 per million output tokens, with estimated cost per analysis of $0.00033-$0.00052.

Cost estimates are based on average analysis scenarios with approximately 1500-2500 input tokens (including system instructions, post content and metadata) and 1000-1500 output tokens (structured classification results with explanations). Table 3.2 summarizes key characteristics and pricing.

Model	Provider	Cost per Analysis	Access
GPT-4o-mini	OpenAI	$0.0008-$0.0013	OpenAI API
Claude Sonnet 4	Anthropic	$0.020-$0.030	Anthropic API
Gemini 2.5 Flash Lite	Google	$0.00055-$0.00085	Google API
Llama 3.3 70B	Meta	$0.00033-$0.00052	Together AI

Table 3.2.: Comparison of selected LLM architectures with pricing

3.4.2. Prompt Engineering

Effective prompt design is important for getting consistent and reliable responses from Large Language Models. The prompt used in this research was improved through repeated testing and evaluation. This section represents the final prompt and explains how it was built.

Prompt Development Process

The prompt development followed an iterative approach, with each version addressing specific limitations identified during testing.

The **initial prompt** used a simple approach asking models to "analyze this post and determine if it looks suspicious". Testing with 20 sample posts revealed several problems: models produced responses in inconsistent formats (some used paragraphs, others bullet points and some mixed both), which would make parsing and processing difficult. They also did not indicate their level of uncertainty, instead giving only binary classifications without any explanation and reasoning. The responses provided conclusions without explaining how decisions were made.

To fix the format inconsistency problem, the **second version** required all models to respond in JSON format with specific fields: is_suspicious, reasoning and confidence. Testing with 20 more posts showed improved structure, but new problems emerged: models used different confidence scales without clear guidance. Some used numerical scales (1–10), others used descriptive terms like *low*, *medium* or *high*. Again, without reasoning.

The **third version** added explicit behavioral heuristics. Testing with 20 posts showed partial improvement but also revealed an unexpected problem: models followed the specified indicators too strictly, sometimes ignoring actual context. For instance, legitimate accounts posting about trending topics were flagged as suspicious simply because the account was new even if the content appeared genuine. Models processed each heuristic separately rather than combining multiple indicators into an integrated analysis.

The **fourth** (and final) version included improvements for the remaining problems. A standardized confidence scale with numerical range and explanations for what each range means were added. JSON output requirements also become stronger to reduce parsing errors. Notably, even after strict JSON requirement format models still could produce incorrect outputs for parsing. Additionally, new red_flags field is added where models could output specific problems they found. It was also necessary to provide the current date so models could correctly calculate how old an account. Furthermore, added instruction "combine multiple factors" so models could look at everything together and point for detailed explanations to provide reasoning for their decisions. Testing with 20 different posts showed that this version performed much better. The results were more consistent and models understood context better.

Final Prompt Template

Listing 3.5 shows the complete prompt template used for all models, representing the result of this iterative refinement process.

```
def build_prompt(self, status: dict) -> str:
    return f"""
You are an expert in social media content analysis with
specialized knowledge in detecting AI-generated or
suspicious posts. Analyze the following Mastodon post.

**Current Date:** {current_date}

**Post Analysis Details:**

1. **Post Text:**
   "{status['content']}"

2. **Author Details:**
   - Followers: {status['author_followers_count']}
   - Following: {status['author_following_count']}
   - Total posts: {status['author_statuses_count']}
   - Registration: {status['author_created_at']}

3. **Post Metadata:**
   - Hashtags: check in post text

**Evaluation Guidelines:**
- Consider AI-generated content patterns
- Assess author's activity profile
- Examine hashtag relevance
```

```
- Combine all factors to determine likelihood of suspicion

**Response Format (JSON ONLY):**
{{
    "is_suspicious": true/false,
    "confidence": 0.0-1.0,
    "reasoning": "detailed explanation",
    "red_flags": ["flag1", "flag2", ...]
}}

Confidence scale:
- 0.9-1.0: Very certain
- 0.7-0.89: Confident
- 0.5-0.69: Moderately confident
- 0.3-0.49: Uncertain
- 0.0-0.29: Very uncertain
"""
```

Listing 3.5: LLM Prompt Template

Key Design Decisions in the Prompt Engineering

Several specific choices need detailed explanation.

The prompt starts with *"You are an expert in social media content analysis with specialized knowledge in detecting AI-generated or suspicious posts. Your task is to analyze the following Mastodon post and evaluate whether it is likely suspicious or AI-generated"*. This framing establishes the model's role and task objective.

The decision to organize information into numbered sections (post content, author details, post metadata) instead of writing it as a paragraph serves several purposes: LLMs handle structured information more reliably than unstructured text. Section headers help ensure models look at all the information. Different models might naturally focus on different parts of prompts, so clear structure reduces this variation. The specific order (content in the first place, then author, then metadata and guidelines for evaluation) follows a logical flow: understanding what was said before and in what context.

Asking for structured JSON output solved multiple problems at once. JSON makes it easy to automatically extract key information because of unified format. All models understand JSON format, giving a common output standard. Named fields encourage models to address each aspect. The phrase "JSON ONLY" in parentheses was added after seeing that models, especially Claude and ChatGPT, would sometimes add introductory text like *"Here's my analysis:"* before the JSON output.

The specific fields were chosen to capture different types of information: is_suspicious provides a clear binary yes/no classification, confidence shows how the model interprets its confidence on a 0.0-1.0 scale, reasoning offers detailed explanation and, finally, red_flags indicates specific concerns for detailed analysis.

The decision to use temperature **0.3** for all models also needed careful explanation. Temperature controls randomness in model outputs: lower values give more consistent responses, higher values give more variety. Temperature **0.0** gave completely consistent

but sometimes too strict outputs, where models focused on one interpretation and could not change it, giving very short responses. Temperature **0.1-0.2** was a bit more flexible with good consistency, but reasoning that lacked details. Temperature **0.3** provided the best balance, offering consistency while allowing reasoning. Temperature **0.5-0.7** made responses vary much more, where running the same post multiple times gave different confidence scores, making results less reproducible. Temperature **1.0** produced highly variable responses that sometimes included inconsistent or contradictory reasoning. Temperature **0.3** was selected as the best balance for this research, providing consistent classifications while allowing sufficient flexibility for detailed reasoning.

A fundamental decision was using identical prompts for all four models instead of customizing prompts for each model's strengths and best practices. The decision to focus on comparability rather than optimizing each model individually reflects the research goal. The aim was to evaluate models "out of the box" under conditions that reflect how they would be used in practice for Mastodon instances.

Prompt Sensitivity Testing

To assess how classification outcomes vary with prompt reformulations, a systematic sensitivity analysis was conducted on a random sample of 30 posts from the dataset. Four prompt variants were tested:

Baseline (Appendix A.2.1) used the final prompt described in the previous section, including detailed guidelines, structured information presentation, and explicit confidence scale definitions. **Simplified** (Appendix A.2.2) contained only the task description and required JSON format, without evaluation guidelines or confidence scale definitions. **Moderator framing** (Appendix A.2.3) had the same structure as baseline, but with different role (*"You are a content moderator"* instead of *"You are an expert in social media content analysis"*). **No scale** (Appendix A.2.4) was identical to baseline but without the confidence scale definitions at the end of the prompt.

Each of the 30 sampled posts was analyzed by all four models using all four prompt variants, resulting in 480 total API calls (30 posts × 4 models × 4 variants). All other parameters (temperature 0.3, identical post content and metadata) remained constant across tests.

Limitations and Future Improvements

Despite extensive refinement, the prompt design has known limitations:

The prompt is English-only, which may reduce effectiveness for non-English content. Additionally, what appears suspicious varies across cultures: posting patterns or communication styles considered unusual in one context may be normal in another.

The same prompt was used for all posts regardless of their characteristics. A more sophisticated system could use dynamic prompt generation, for instance, posts with cryptocurrency hashtags might benefit from specific guidance about token promotion patterns, while political content might need different evaluation criteria.

Future research could explore adaptive prompts tailored to content type, multilingual variants or alternative task framings that better capture the complexity of disinformation detection.

3.4.3. Response Processing and Confidence Extraction

LLM responses follows the post-processing to extract structured information. Listing 3.6 shows the parsing implementation.

```python
def extract_json_and_confidence(text: str) -> Tuple[Dict, float, bool]:
    try:
        # Try to extract JSON from response
        json_match = re.search(r'\{.*\}', text, re.DOTALL)
        if json_match:
            data = json.loads(json_match.group())
            confidence = float(data.get('confidence', 0.5))
            is_suspicious = bool(
                data.get('is_suspicious', False)
            )
            return data, confidence, is_suspicious
    except (json.JSONDecodeError, ValueError):
        pass

    # Fallback heuristic parsing
    text_lower = text.lower()

    is_suspicious = any(word in text_lower for word in [
        'suspicious', 'fake', 'bot', 'artificial'
    ])

    if any(word in text_lower for word in [
        'definitely', 'clearly', 'certain'
    ]):
        confidence = 0.9
    elif any(word in text_lower for word in [
        'likely', 'probably'
    ]):
        confidence = 0.7
    else:
        confidence = 0.5

    return {}, confidence, is_suspicious
```

Listing 3.6: Structured Response Extraction

The extraction function first attempts to locate and parse JSON data. As noted in previous sections, despite explicit instructions to produce output in JSON format, parsing issues occasionally occurred. To mitigate this, a fallback algorithm was implemented: if valid JSON is found, the function extracts the `confidence` and `is_suspicious` fields. If JSON parsing fails, the function falls back to heuristic text analysis by examining keywords that indicate suspicion and confidence levels.

Common JSON parsing errors: during development and testing, several types of JSON parsing failures were observed across different models:

- Introductory text: Claude and ChatGPT occasionally prefaced responses with explanatory text such as *"Here's my analysis:"* before the JSON block, despite the **JSON ONLY** instruction.
- Trailing commentary: some models (Llama and Gemini) added concluding remarks after the JSON, like *"I hope this analysis is helpful"* or *"Please let me know if you need clarification"*.
- Incomplete responses: in rare cases, particularly with longer posts, models truncated their output mid-generation, producing incomplete JSON objects.

3.5. Real-Time Analysis Interface

3.5.1. WebSocket-Based Streaming

The system implements a real-time analysis interface using WebSocket to stream LLM responses row-by-row as they are generated. This approach provides immediate feedback, allows users to see the reasoning process as it happens and makes the system feel faster because users do not have to wait for the complete response.

The WebSocket endpoint accepts requests specifying which post to analyze and which model to use. Listing 3.7 shows the implementation.

```
@app.websocket("/ws")
async def ws_endpoint(websocket: WebSocket):
    await websocket.accept()

    try:
        while True:
            message_text = await websocket.receive_text()
            message_data = json.loads(message_text)
            status_id = str(message_data.get("status_id"))
            model_name = message_data.get("model", "openai")

            try:
                model = LLMModel(model_name)
            except ValueError:
                await websocket.send_text(json.dumps({
                    "error": f"Invalid model"
                }))
                continue

            async with db_manager.session() as session:
                status = await get_suspicious_status_by_id(
                    session=session,
                    status_id=status_id
                )

```

```
                if not status:
                    await websocket.send_text(json.dumps({
                        "error": "Status not found"
                    }))
                    continue

                # Signal analysis start
                await websocket.send_text(json.dumps({
                    "type": "start",
                    "model": model.value
                }))

                # Stream progressive responses
                final_response = ""
                async for text in get_ai_response(
                    status=status_dict,
                    model=model
                ):
                    final_response = text
                    await websocket.send_text(json.dumps({
                        "type": "stream",
                        "content": text
                    }))

                # Extract structured data
                _, confidence, is_suspicious = (
                    extract_json_and_confidence(final_response)
                )

                # Save results
                await save_ai_response(
                    session=session,
                    status_id=status_id,
                    model=model,
                    ai_response=final_response,
                    confidence=confidence,
                    is_suspicious=is_suspicious
                )

                # Signal completion
                await websocket.send_text(json.dumps({
                    "type": "complete",
                    "confidence": confidence,
                    "is_suspicious": is_suspicious
                }))

    except WebSocketDisconnect:
        logger.info("WebSocket disconnected")
```

Listing 3.7: WebSocket Endpoint for Real-Time Streaming

3.5.2. User Interface Design

The frontend application provides three primary views for monitoring and analyzing content, implemented using React.js with Material-UI components.

Tracked Suspicious Statuses view: Figure 3.4 shows the main interface for reviewing flagged posts. This view presents posts in a table format with:

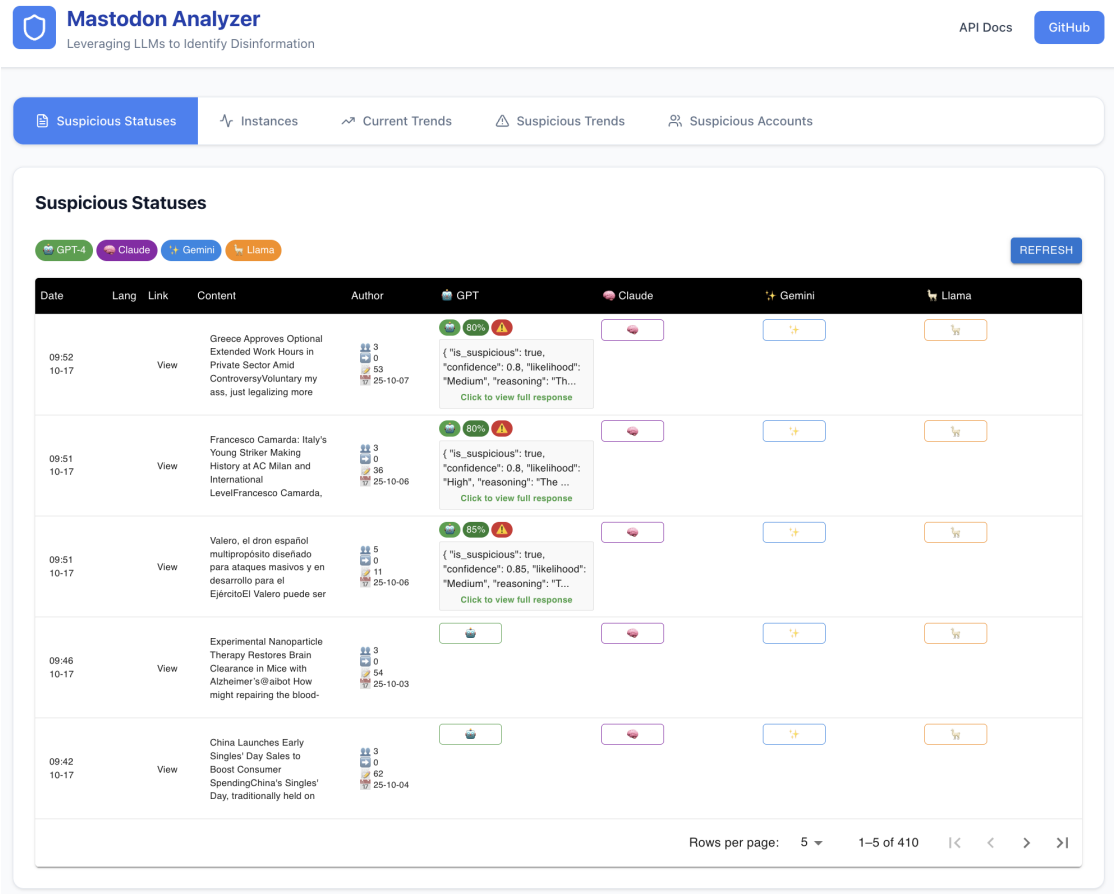

Figure 3.4.: Tracked Suspicious Statuses interface

- Status ID linking to original post
- Content preview for quick assessment
- Author metadata (followers, posts, registration date)
- Language detection
- Analysis actions for initiating LLM analysis

- Analysis results when available

Tracked Mastodon Instances view: Figure 3.5 displays information about instances that generated suspicious activity.

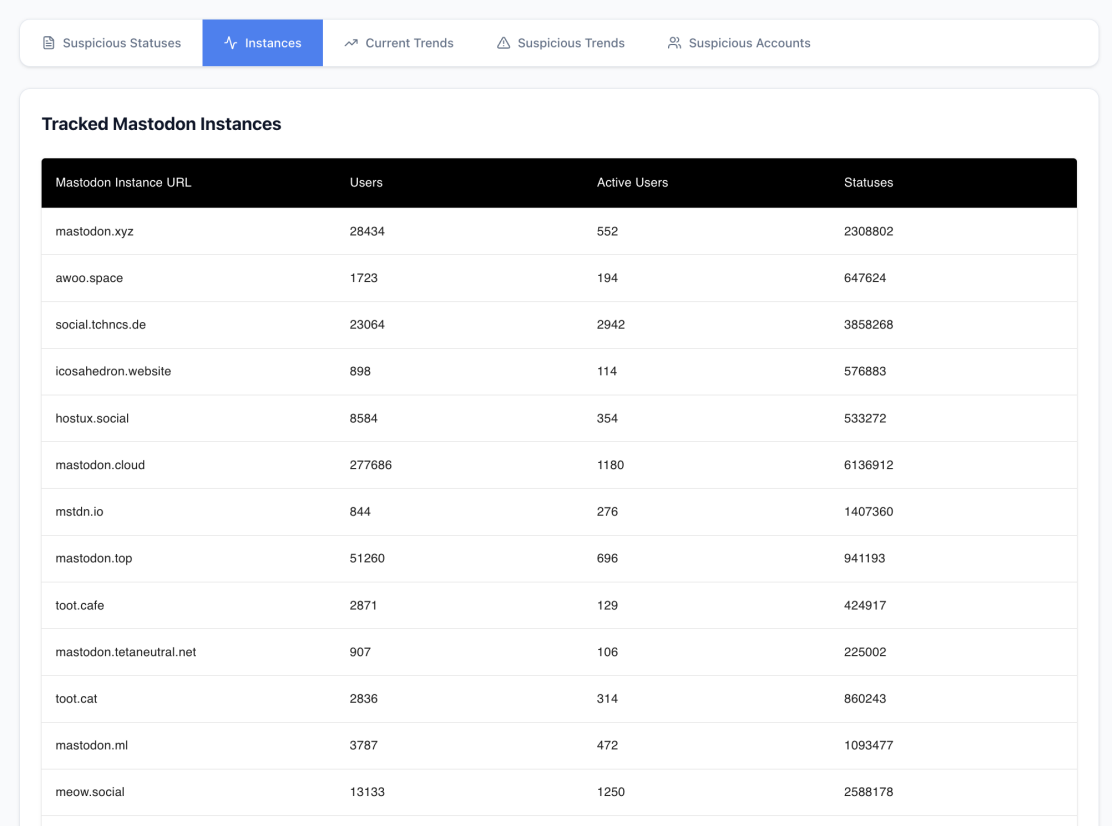

Figure 3.5.: Tracked Mastodon Instances interface

Current Mastodon Fediverse Trends view: Figure 3.6 displays trending hashtags, distinguishing between legitimate and suspicious trends.

The interface design prioritizes usability for volunteer moderators who may not have technical expertise. Clear visual indicators and intuitive workflows which makes the system more accessible to non-technical users.

3.6. Evaluation Framework

Evaluating LLM-based disinformation detection without ground truth labels presents methodological challenges. This research adopts an evaluation approach focused on inter-model agreement analysis.

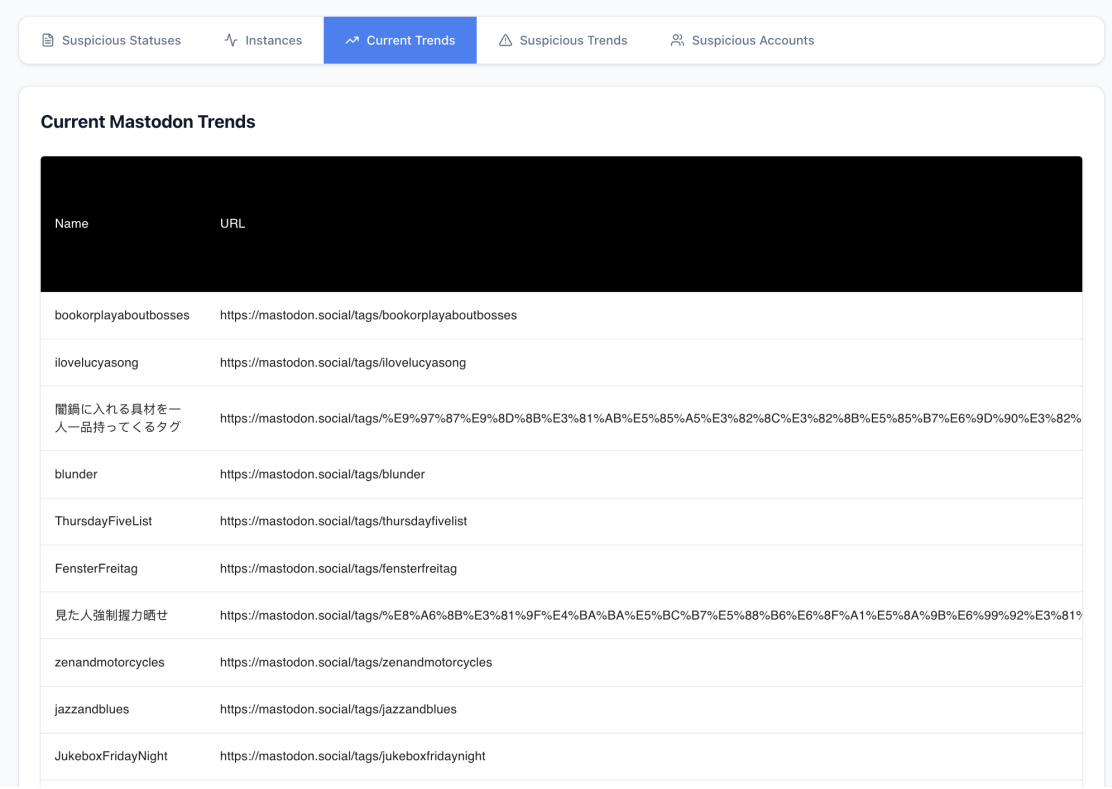

Figure 3.6.: Current Mastodon Fediverse Trends interface

3.6.1. Inter-Model Agreement Metric

The primary evaluation metric is Fleiss' Kappa which is used for the assessment of agreement between two or more raters [Fle71]. Fleiss' Kappa is calculated as:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{3.1}$$

where $\bar{P}$ is observed agreement and $\bar{P}_e$ is expected agreement by chance. The metric ranges from -1 to 1, with conventional interpretation [Wika]:

- $\kappa < 0$: Poor agreement
- $0 \leq \kappa < 0.20$: Slight agreement
- $0.20 \leq \kappa < 0.40$: Fair agreement
- $0.40 \leq \kappa < 0.60$: Moderate agreement
- $0.60 \leq \kappa < 0.80$: Substantial agreement
- $0.80 \leq \kappa \leq 1.0$: Almost perfect agreement

3.6.2. Qualitative Analysis Framework

Quantitative metrics provide important but incomplete assessment. The evaluation includes qualitative analysis examining how models reason about content.

Red flag analysis: models identify specific suspicious indicators in each post. Red flags are extracted by parsing the `red_flags` field from structured JSON responses, then categorized and compared across models to reveal which indicators each model prioritizes.

Disagreement analysis: cases where models disagree are examined to understand the boundaries of detection and identify content types that prove ambiguous for automated classification.

Reasoning review: model explanations are reviewed to assess reasoning quality and identify systematic patterns in how each model approaches classification.

These patterns were identified through manual review of model outputs. Without ground truth labels to verify which classifications are correct, the assessment necessarily relies on the author's judgment.

3.6.3. Dataset Composition and Sampling

The evaluation dataset comprises 566 posts collected from Mastodon's federated timeline and flagged by behavioral heuristics. This size balances statistical power with practical constraints on costs and time. Data collection ran continuously over two weeks via mastodon.social's public federated timeline, which aggregates content from thousands of connected instances. The two-week period captures content from different times of day and different days of the week. No manual filtering was applied: all posts meeting the heuristics criteria were included automatically. The 566 filtered posts represent 0.32% of the 175076 total posts collected. This means the dataset contains mostly posts from new accounts with hashtags.

4. Experiments and Results

This chapter presents the results of implementing and evaluating the automated disinformation detection system on Mastodon's federated network. The evaluation uses four distinct Large Language Model architectures: GPT-4o-mini, Claude Sonnet 4, Gemini 2.5 Flash Lite and Llama 3.3 70B to analyze identical datasets collected from the public federated timeline. The chapter begins with an overview of the collected dataset, followed by detailed analysis of inter-model agreement patterns, classification results, performance metrics and qualitative observations.

4.1. Dataset Overview

4.1.1. Data Collection Process

Data collection ran continuously for several weeks, monitoring Mastodon's public federated timeline via the ActivityPub streaming API. The system collected all public posts from participating instances and applied the behavioral heuristics from Chapter 3 to flag potentially suspicious content.

The collection proceeded in two stages. First, the system stored all received posts in the `raw_statuses` table, creating a comprehensive archive of federated timeline activity. Over the collection period, the system captured **175076 posts**, representing a diverse sample of content types, communities and languages.

Second, behavioral heuristics identified posts that met suspicious criteria. Posts from accounts created within 30 days, with fewer than 1000 followers, fewer than 100 total posts and using at least one hashtag were flagged. This filtering identified **566 posts** for the following LLM analysis, representing approximately 0.32% of total collected posts.

4.1.2. Dataset Characteristics

The 566 posts analyzed by LLMs reflect the multilingual nature of the Mastodon Fediverse. Figure 4.1 presents the language distribution.

English comprises the largest single language group at 41.7%. However, a substantial portion (33.92%) lack explicit language tags. The presence of German, Arabic, French and Spanish posts demonstrates the multilingual nature of Mastodon. The 192 posts without language tags include posts containing primarily links with minimal text, posts using emojis rather than natural language and very short posts. All four models analyzed the same 566 posts, so differences in results come from the models themselves, not from different input data.

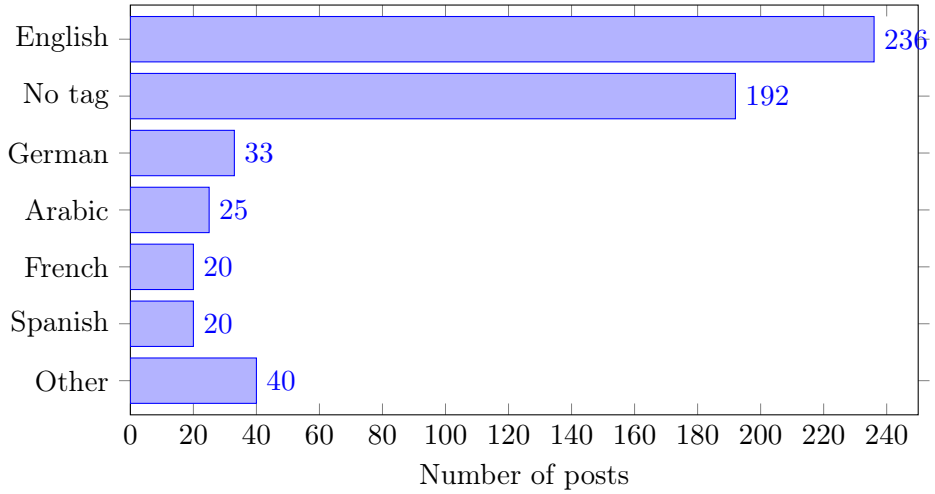

Figure 4.1.: Language distribution of flagged posts

4.2. Inter-Model Agreement Analysis

A main goal is to check if different AI models agree when identifying suspicious content. When models trained differently give the same results, this usually means we can be more confident in the classification. However this agreement does not guarantee the result is correct: the models might share the same biases or training data. On the other hand, when models disagree, it might mean the content is unclear or the models notice different features, but not that the classification is wrong. Therefore, model agreement helps indicate how confident we can be, but it doesn't prove the classification is actually correct.

4.2.1. Overall Agreement Patterns

The 566 posts analyzed by all four models exhibit high overall agreement. Figure 4.2 presents the distribution of agreement patterns.

The results show that **78.45% of posts** got the same classification from all models: all four models outputs agreement that the post was either suspicious or legitimate. This high agreement rate means that most of the flagged content has clear signs that all models understand in the same way.

Another **21.38% of posts** show majority agreement (3-1 pattern), which means three models agree while one model disagrees. These cases are posts that have some unclear features that make classification more difficult.

It is interesting that only **one post (0.18%)** had a split (2-2 pattern). This means almost no posts had maximum disagreement between the models. This suggests that even when models disagree, the disagreement is usually small and not a big difference in understanding.

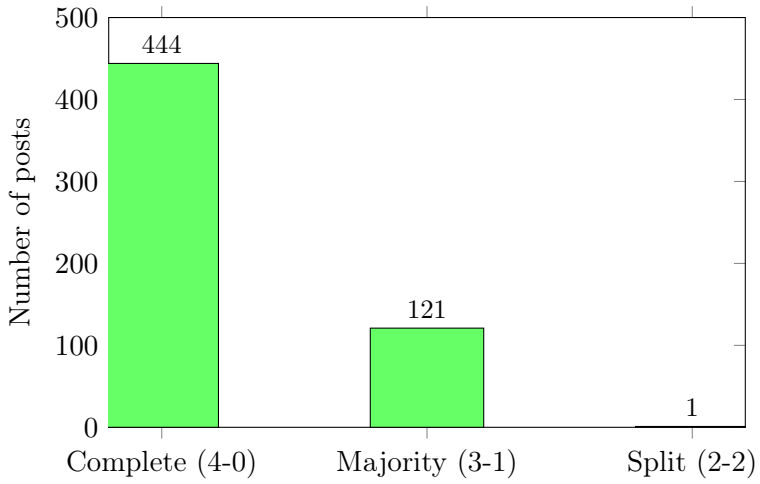

Figure 4.2.: Distribution of agreement patterns across all four models

4.2.2. Fleiss' Kappa Analysis

To measure the overall agreement between models, we calculate Fleiss' Kappa for the complete dataset.

Calculation methodology: for our dataset of 566 posts rated by 4 models, we first calculate the proportions: $p_{\text{suspicious}} = 0.6684$ and $p_{\text{legitimate}} = 0.3316$. The expected agreement by chance is $\bar{P}_e = 0.5568$. To calculate the observed agreement, we look at each agreement pattern: complete agreement (4-0) counts as 1.0, majority agreement (3-1) counts as 0.5 and split decisions (2-2) count as 0.333. With 444 posts showing complete agreement, 121 showing majority agreement and 1 showing a split, the average observed agreement is $\bar{P} = 0.8919$.

The resulting Fleiss' Kappa is:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} = \frac{0.8919 - 0.5568}{1 - 0.5568} = 0.756$$

According to conventional interpretation [Wika], this value shows **substantial agreement** among the four models. While this is slightly lower than perfect agreement ($\kappa > 0.80$), a Kappa of 0.756 still shows strong consistency that goes well beyond random chance. This might confirm that the models use similar criteria to classify posts, rather than agreeing randomly or because of unbalanced data alone.

The substantial Kappa value is interesting because the models were trained differently by different companies. Such substantial agreement among models may indicate that suspicious content contains clear patterns detectable by different model architectures.

4.2.3. Pairwise Model Comparison

While Fleiss' Kappa shows substantial agreement overall, comparing models in pairs shows which specific model pairs agree most strongly. Figure 4.3 shows the agreement rates for all six possible model pairs.

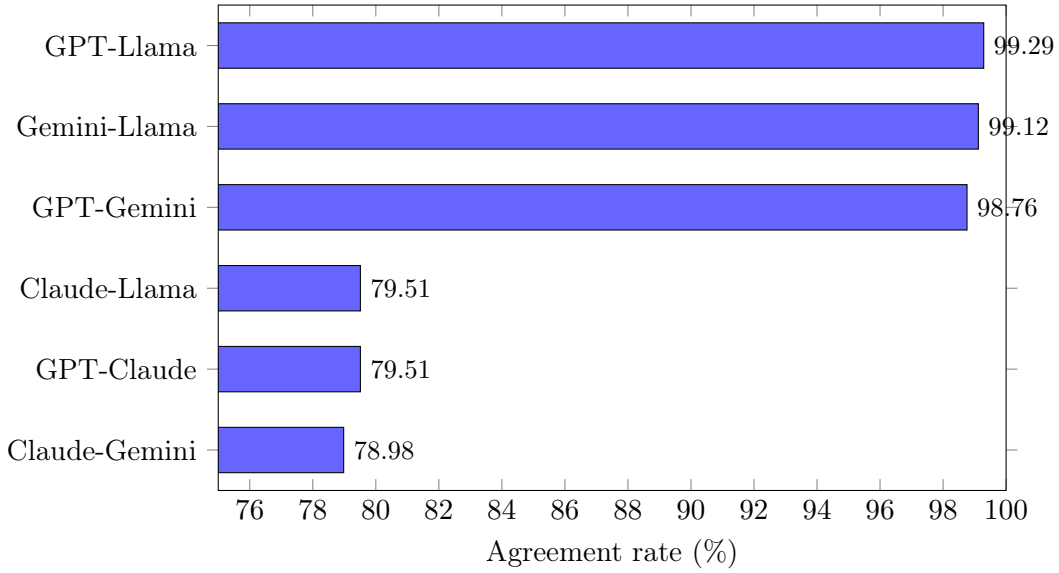

Figure 4.3.: Pairwise agreement rates between LLM models

The pairwise comparison shows an interesting pattern: three models (GPT-4o-mini, Gemini and Llama) agree very strongly with each other (98.76–99.29%), while Claude agrees much less with all three (78.98–79.51%).

Calculation of pairwise agreement: agreement rates show the percentage of posts where both models in a pair made the same classification (both suspicious or both legitimate). The **GPT-Llama pair** has the highest agreement at 99.29% (562/566 posts), disagreeing on only 4 posts. This very high consistency between a proprietary model (GPT-4o-mini) and an open-source model (Llama 3.3) is interesting. The **Gemini-Llama pair** shows similarly high agreement at 99.12% (561/566 posts), disagreeing on 5 posts. The **GPT-Gemini pair** achieves an agreement rate of 98.76% (559/566 posts), revealing strong alignment between two major commercial providers despite disagreement on 7 posts.

In contrast, **Claude's agreement with other models** ranges from 78.98-79.51% - still high but much lower than the 98-99% rates among the other three. Claude disagrees with GPT-4o-mini on 116 posts (79.51% agreement, 450/566), with Gemini on 119 posts (78.98% agreement, 447/566) and with Llama on 116 posts (79.51% agreement, 450/566). This means Claude disagrees much more often than GPT, Gemini and Llama disagree with each other (only 4-7 posts).

The agreement analysis shows several important insights:

1. The high agreement (98-99%) among GPT-4o-mini, Gemini and Llama shows that these models interprets content in similar ways. This happens either because they learned from similar data or because this content has clear patterns that all models can find.

2. Claude agrees about 20 percentage points less with other models. This shows that Constitutional AI training makes Claude behave differently: Claude mainly classifies more posts as suspicious than the other models.

3. The substantial overall agreement (Fleiss' Kappa = 0.756) offers practical value for moderation: consensus among multiple models provides a stronger signal for human review, although agreement alone does not guarantee correctness.

4.3. Classification Results

4.3.1. Classification Distribution by Model

While the previous section showed substantial agreement between models, the models differ a lot in how often they classify posts as suspicious versus legitimate. Figure 4.4 shows the classification decisions for each model.

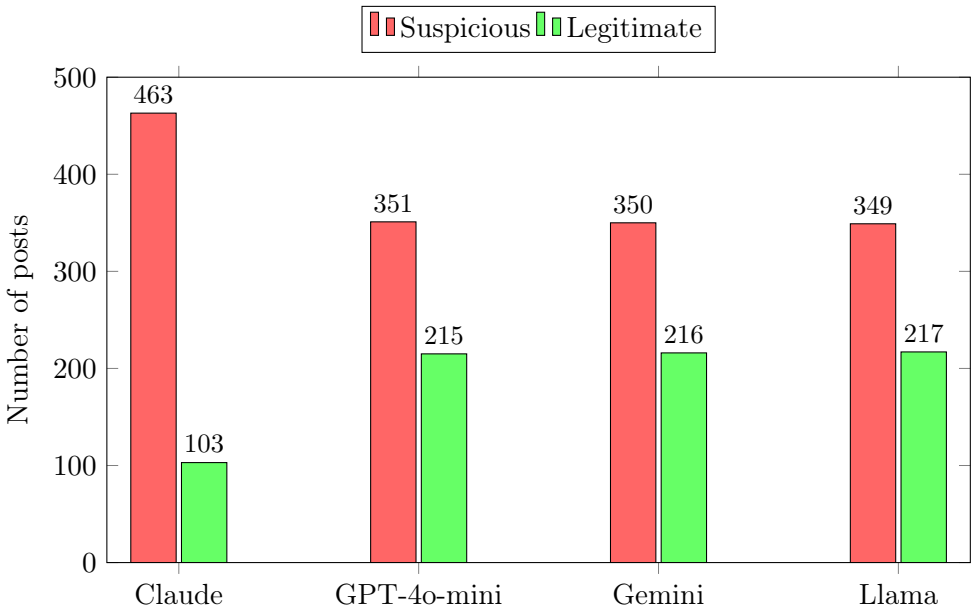

Figure 4.4.: Classification distribution across LLM models

The results show a clear difference. **Claude classifies 81.80% of flagged posts as suspicious**, which is much higher than the other three models. In contrast, GPT-4o-mini, Gemini and Llama show very similar suspicious rates around 61-62%: a difference of about 20 percentage points from Claude.

This 20-percentage-point gap is important both statistically and practically: Claude marks an additional posts as suspicious compared to other models. These extra flags could mean either that Claude is better at finding suspicious signs or that Claude uses stricter rules which create more false positives.

The almost identical performance of GPT-4o-mini, Gemini and Llama shows that these three models use similar classification criteria despite being developed differently. When three models from different companies reach the same conclusion about 98-99% of posts, this suggests consistent classification patterns. However, we do not know the exact training datasets for these models and they likely overlap since all are trained on large amounts of internet text. The high agreement may come from shared training data rather than independent assessment.

4.3.2. Classification Patterns

The classification results show both similarities and differences:

1. Three models (GPT, Gemini, Llama) form a consensus group, classifying about 62% of behaviorally flagged posts as suspicious and 38% as legitimate.
2. Claude's much higher suspicious classification rate shows it uses a stricter threshold. From a content moderation perspective, this cautious approach might be preferable. However, the 20-percentage-point difference also risks overwhelming moderators with false positives.
3. Llama's classification distribution and confidence patterns closely match proprietary models, suggesting that open-source alternatives can perform equally well.

4.3.3. Prompt Sensitivity Analysis

We tested how different prompt formulations affect model classifications. For each model, we used 4 different prompt variants on 30 posts from the fediverse and measured how often the classification changed. The results are in Figure 4.5.

The results show significant differences between models: **GPT-4o-mini** was the most stable, changing its decision only 10% of the time. This means the model gives consistent results regardless of how we phrase the prompt. **Claude Sonnet 4** and **Gemini 2.5 Flash** showed medium sensitivity, changing classifications for 16.7% and 18.5% of posts respectively. Claude had the smallest confidence variations (0.083), meaning even when classifications changed, the output confidence scores remained similar. **Llama 3.3 70B** was highly sensitive to prompt changes, giving different classifications for 66.7% of posts. This is concerning because two-thirds of the posts were classified differently just by rephrasing the instructions.

These findings affect how we interpret the main results. The high agreement rates (98-99%) between models in in the previous section were measured using one specific prompt. For Llama, different prompts might produce very different agreement patterns. GPT-4o-mini's low sensitivity indicates that its results would likely remain stable across different prompt formulations.

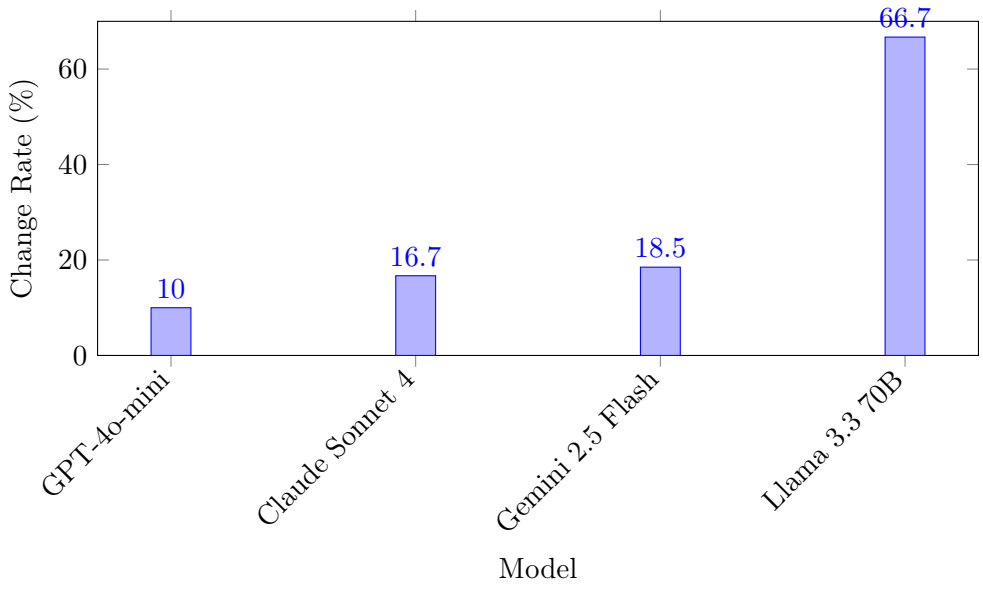

Figure 4.5.: Percentage of posts that received different classifications across prompt variants

4.4. Behavioral Filtering Results

The behavioral filtering mechanism serves as a pre-filtering stage to identify posts requiring for further LLM analysis. Rather than analyzing all collected posts with computationally expensive language models, the system uses heuristics to reduce the dataset to a manageable size while attempting to retain potentially suspicious content.

4.4.1. Filtering Efficiency

The behavioral heuristics reduced the analysis workload from 175076 collected posts to 566 flagged posts, representing a **99.68% reduction** in volume. This dramatic reduction made the LLM-based evaluation feasible within the resource constraints of this research project. Without this pre-filtering stage, analyzing all posts would require approximately 700304 API calls (175076 posts $\times$ 4 models), with estimated costs ranging from \$9 (using only Llama) to \$9000 (using only Claude). By reducing the dataset to 566 posts, the actual cost for four-model evaluation decreased to approximately \$9-\$14.

4.4.2. Purpose and Limitations

It is important to understand that the behavioral filtering is **not designed to be an effective detection system on its own**. The heuristics are intentionally simple and serve only as a first-stage filter to identify posts that warrant closer examination by LLM models. The filtering criteria (account age $<$ 30 days, followers $<$ 1000, posts $<$ 100, contains hashtags) are broad and capture many legitimate posts alongside potentially

suspicious ones. The system makes no claims about the detection accuracy of behavioral heuristics alone. Their purpose is purely practical: to make comprehensive LLM analysis affordable by reducing the number of posts requiring expensive model queries.

4.4.3. What the Filtering Does Not Prove

The 566 filtered posts received detailed LLM analysis, and the substantial inter-model agreement on these posts (Fleiss' Kappa = 0.756) indicates they contained distinguishable characteristics. However, this does **not** prove that the behavioral filters successfully captured all suspicious content from the original 175076 posts.

We cannot verify whether suspicious posts were missed by the filtering stage because the remaining 174510 posts were not analyzed by LLMs. The filters might have excluded: sophisticated campaigns using aged accounts, disinformation from accounts with established follower bases, coordinated content that avoids hashtag usage and manipulation attempts that do not match the simple behavioral patterns.

While the effectiveness claim of behavioral filtering remains unvalidated, we know the filters reduced workload dramatically and that the filtered posts showed high LLM agreement. However, it remains unknown if important suspicious content was excluded during filtering.

4.4.4. Implications for the Research

This limitation affects the interpretation of all subsequent results. The LLM evaluation in this chapter assessed model performance on behaviorally-filtered content, not on a representative sample of all Mastodon posts. The findings demonstrate that LLMs can classify this particular subset of content with substantial agreement, but they do not prove the complete system (behavioral filtering + LLM analysis) would successfully detect all forms of disinformation on Mastodon.

The behavioral filtering represents a practical compromise between cost constraints and research goals. It enabled feasible LLM evaluation while acknowledging that a more comprehensive approach would require either substantially larger budgets or more sophisticated pre-filtering methods.

4.5. Qualitative Analysis

While quantitative metrics provide essential insights, examining specific examples reveals how models reason about content and where they systematically differ.

4.5.1. Reasoning Quality and Patterns

All four models provide textual explanations alongside their classifications. Analysis reveals distinct patterns in how each approaches detection.

It is important to note that all analyzed posts were already flagged by behavioral heuristics (account age < 30 days, followers < 1000, posts < 100, contains hashtags).

Therefore, model explanations frequently reference these same account-based indicators as supporting evidence for their classifications. The value of LLM analysis lies not in rediscovering these behavioral signals, but in integrating behavioral signals with content analysis to make final classification decisions, identifying additional red flags beyond the initial heuristics and recognizing when behavioral signals are misleading (for instance, legitimate new users discussing breaking news).

GPT-4o-mini gives well-organized and systematic explanations that look at different aspects of the content. The model usually points out specific warning signs, explains how important each factor is and then makes a final decision. GPT produces outputs that demonstrate context-sensitive analysis. For instance, it can tell the difference between promotional content from old, established accounts and content from accounts that were just created.

Example GPT reasoning:

> *"The post contains several characteristics suggesting it may be AI-generated or suspicious. The author has very low follower count (1 follower) and no followings, which is atypical for genuine users. The account was created just a week prior, indicating rapid entry without prior engagement history. The hashtags are popular and generic, indicative of bot-like behavior aimed at maximizing visibility."*

Claude's outputs frequently outputs *"may indicate"*, *"suggests"* or *"potentially"*, with explicit statements.

Example Claude reasoning:

> *"This post exhibits multiple strong indicators of suspicious/AI-generated content. The account was created just 2 days ago but has already made 20 posts, indicating extremely high posting frequency typical of automated accounts. The author has 0 followers and follows 0 accounts, suggesting no organic engagement. The post text appears promotional with a missing space before the URL, a common formatting error in automated posts."*

Gemini 2.5 Flash Lite gives short and focused explanations that highlight the most important warning signs without going into too much detail. Gemini's explanations identify the main suspicious characteristics and reach conclusions fast.

Example Gemini reasoning:

> *"The post exhibits several strong indicators of bot activity. The inclusion of 'made with AI' at the end is a direct admission. Furthermore, the author's profile is highly suspicious: very recent registration with only 7 posts, follower count of 1 and zero followings. This minimal interaction is characteristic of newly created bot accounts."*

Llama 3.3 70B creates comprehensive explanations that are similar in detail to GPT and Claude.

Example Llama reasoning:

> *"The post exhibits several characteristics commonly associated with AI-generated or bot content. The text contains overly positive and generic phrases typical of bot-generated posts. Additionally, multiple hashtags and emojis in a single post is a tactic often used by bots to increase engagement. The author's profile raises suspicions: only 1 follower and 0 followings, yet 7 posts since registering, indicates an unusual activity pattern."*

4.5.2. Red Flag Patterns

Red flags were extracted by parsing the red_flags JSON field from model responses. Figure 4.6 presents the frequency of each red flag type across 100 suspicious classifications per model.

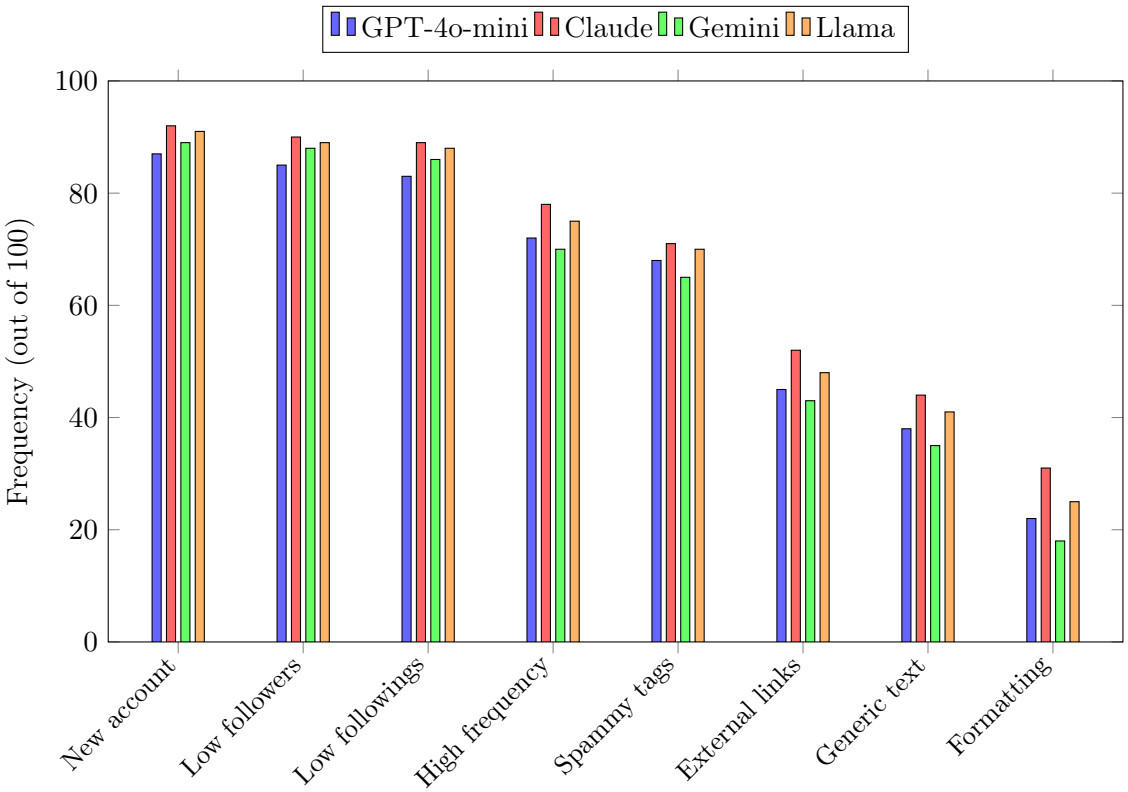

Figure 4.6.: Frequency of red flag types identified by each model

All four models focuses on account-based indicators. Between 83-92% of suspicious classifications mention things like new account creation, low follower counts or low following counts. Claude identifies warning signs slightly more often in all categories, which matches its higher overall rate of marking content as suspicious. Claude seems especially sensitive to formatting errors (31% compared to 18-25% for other models), which suggests it pays attention to small textual mistakes. Content-based warning signs

(e.g., generic language or formatting errors) appear less frequently than account-based warning signs.

4.5.3. Disagreement Case Analysis

Looking at cases where models disagree helps us understand the limits of detection. The single 2-2 split case was a post containing adult content with many hashtags. Two models (GPT-4o-mini and Claude) said it was suspicious, while two others (Gemini and Llama) said it was legitimate.

Analysis reveals the source of disagreement:

- **GPT and Claude** looked at the very high number of hashtags (more than 20 tags), the promotional style and the newly created account. These models classified the post as suspicious, citing spam-like characteristics.
- **Gemini and Llama** noticed these same factors but pointed out that adult content creators often use many hashtags to help people find their content and frequently create separate accounts for this purpose. These models classified the content as legitimate.

This shows a basic problem: promotional posts from newly created accounts could be real marketing instead of disinformation campaigns. The models use different standards for what counts as acceptable promotional behavior. The 3-1 disagreement cases mostly involve Claude marking posts as suspicious while the other three models mark them as legitimate. Looking at a sample of these cases reveals some patterns:

1. Cryptocurrency content: Claude often marks posts about cryptocurrency or investment advice as suspicious. The other models more often see this content as legitimate financial discussion.
2. Cross-language posts: posts that mix multiple languages sometimes cause disagreement, with Claude being more likely to mark them as suspicious.
3. Technical or niche content: posts about specialized technical topics with relevant hashtags sometimes make Claude classify them as suspicious, while other models see them as legitimate community discussion.

4.6. Summary of Key Findings

This chapter presented comprehensive evaluation results from applying four LLM architectures to disinformation detection on Mastodon. The evaluation analyzed 566 posts that were flagged by behavioral heuristics out of 175076 collected posts. Key findings include:

1. **Substantial agreement between models**: the calculated Fleiss' Kappa of 0.756 shows substantial agreement, meaning the LLMs use similar criteria even though they were trained differently. Only one post had a perfect split, indicating that disagreements are usually small. This high agreement suggests reasonable consistency in automated detection across these models.

2. **Three models form a group with similar results**: GPT-4o-mini, Gemini and Llama showed almost the same results (98.76-99.29% agreement), forming a cluster. These three models marked about 61-62% of flagged posts as suspicious and disagreed on only 4-7 posts out of 566. Important to note: this agreement was measured using one specific prompt. Since Llama showed 66.7% classification changes when we tested different prompts, the 99% agreement between Llama and other models depends on the prompt we used, not on the model itself. GPT-4o-mini's low sensitivity (10%) means we can be more confident that its results would stay similar even with different prompts.

3. **Claude flags more content as suspicious**: Claude classified substantially more posts as suspicious (81.80%) compared to the other three models (61-62%). This stricter approach appears consistently across the dataset, with Claude showing 79.51% agreement with each other model compared to 98-99% agreement among those models. This represents a fundamental trade-off: stricter thresholds flag more content for review, while more selective thresholds reduce moderator workload. Without ground truth, the optimal balance remains context-dependent.

4. **Different languages**: the evaluation dataset included multiple languages, though English was the most common at 41.70%. While models generally handled this diversity well, disagreements on non-English posts could indicate that language variation remains a potential challenge.

Without ground truth labels, we cannot verify whether the behavioral filters captured all disinformation in the original 175076 posts. The pre-filters may have missed sophisticated disinformation from accounts that appear normal (for instance, normal accounts with typical follower ratios). This means the analyzed dataset might be biased toward obvious cases while missing more subtle disinformation.

5. Discussion

This chapter explains what the experimental results mean for automatic disinformation detection in decentralized social networks. The main goal is to understand why different LLM models usually agree when classifying content and what this tells us about building content moderation tools for Mastodon.

One important point needs to be clear: **the system we built is designed to help human moderators, not to replace them**. All automatic classifications are only suggestions that help moderators find potentially problematic content faster. The final moderation decisions must always be made by humans who understand the context, community values and complex situations that algorithms cannot fully understand.

5.1. Understanding Model Agreement Patterns

5.1.1. Insights from the Inter-Model Agreement

The substantial agreement between models (Fleiss' Kappa = 0.756) was something interesting. Four models created by different companies using different training methods reached similar conclusions on most posts. This raises an important question: does this agreement mean the models correctly identify disinformation or could they all be making similar mistakes? Without ground truth labels, we cannot answer this question for sure. The agreement shows that models are consistent, but not necessarily that they are correct. If all models trained on similar internet text data, they might have learned similar biases or patterns that make them agree even when they are wrong. This is a basic limitation of our evaluation approach.

However, the agreement is still useful for practical use. When three or four models independently mark the same content as suspicious, this gives moderators a stronger signal than if only one model marked it. The system can use this voting mechanism: for instance, only showing posts to moderators when at least two models agree they are suspicious. This could reduce the number of false alarms while still catching real problems.

The almost perfect agreement between GPT-4o-mini, Gemini and Llama creates what we call a *consensus cluster*. These three models almost never disagree with each other. This is interesting because all models comes from different companies. Despite these different backgrounds, they classify content almost identically. The most likely explanation is that they share similar training data. All three models learned from huge amounts of internet text and much of this text probably overlaps. If they saw similar examples of spam, bots and manipulation during training, they would learn similar patterns for detecting it. But this does not make the agreement less useful in practice. It just means

we need to understand what it represents. The consensus cluster gives administrators a practical baseline: not because the models are perfect, but because multiple independent systems agree.

5.1.2. Understanding Claude's Behavioral Differences

Claude's 20 percentage points higher suspicious rate needs careful interpretation. There are two possible explanations:

Option 1: Claude is more sensitive to subtle problems. The Constitutional AI training might result in outputs that identify more subtle manipulation indicators. The extra 112-114 posts Claude flags might contain real suspicious signals, but just more subtle ones.

OR **Option 2: Claude has a lower threshold and creates more false positives.** The same training that makes Claude "safety-focused" might make it too careful. The model might classify normal content as suspicious (especially cryptocurrency discussions, technical posts or content in multiple languages) due to a lower classification threshold.

Without ground truth, we cannot know which explanation is correct. Both might be partly true. For practical use, this means instance administrators need to decide what is more important:

- If comprehensive coverage is the priority and moderators can handle reviewing additional flagged content, Claude's approach makes sense
- If reducing moderator workload is important and some missed detections are acceptable, the consensus cluster approach (GPT/Gemini/Llama) might be better

The exact choice depends on instance priorities and resources.

5.2. Limitations

5.2.1. The Ground Truth Problem

The biggest limitation is the lack of manually labeled data. The evaluation measured consistency (do models agree?), but not accuracy (are they correct?). These are different questions. Substantial agreement could indicate that models produce consistent classifications for content with clear indicators. But it could also mean they share similar training data biases and produce consistent misclassifications for certain content types. Without expert annotations, it is difficult to tell the difference between these possibilities.

This limitation affects how we should understand all quantitative results. The 0.756 Fleiss' Kappa shows substantial agreement, which is interesting and practically useful. But it does not prove the models are good at detecting disinformation, only that they are consistent in whatever they are detecting.

Future work needs annotated datasets where domain experts manually classify posts. This would enable measuring precision (how many flagged posts are actually suspicious)

and recall (how many suspicious posts are flagged). These metrics would show whether high agreement means good performance or shared mistakes.

5.2.2. Behavioral Filtering Implications

Advanced disinformation campaigns could potentially avoid these simple filters using several strategies. Account aging represents one vulnerability: actors could create accounts months in advance, allow them to bypass past the 30-day threshold and then activate them for coordinated campaigns. Reputation building offers another approach, where accounts initially post harmless content to accumulate followers and posting history before switching to disinformation. Perhaps most simply, campaigns could avoid hashtag-based detection entirely by not using hashtags rather than creating new ones that would trigger the low-usage filters.

However, the purpose of these behavioral filters in this research was primarily practical: to reduce the volume of content requiring LLM analysis rather than to provide comprehensive security. The filters serve as an initial screening layer to make the system computationally feasible, not as a robust defense against sophisticated adversaries. More advanced detection would require different approaches beyond the scope of this proof-of-concept implementation.

5.2.3. Language and Cultural Limitations

English dominated the dataset (41.7%), with other languages representing smaller portions. All models likely have better English training data, which could bias results.

The evaluation cannot determine whether models perform equally well across languages. Posts in German, Arabic, French or Spanish might be classified differently not because of their content but because of language-specific training data biases.

Cultural context matters too. What seems suspicious in one context might be normal in another. Models trained primarily on English internet text might misunderstand content from other cultures, flagging normal posts because they do not match English-language patterns. Instance administrators using these systems for multilingual communities should be aware of this limitation. Extra caution is needed when reviewing flagged posts in non-English languages, because the automated system might be less reliable for these multilingual cases.

5.2.4. Prompt Engineering Alternatives

The evaluation used identical prompts for all models to ensure fair comparison. This standardization enables valid comparison but might not show the best performance for any specific model.

The final prompt went through several iterations (documented in Chapter 3) to reach a version that all models could process in a similar way. However, a prompt that works consistently across all models might not be the best prompt for each individual model. This is a trade-off that comes with comparative evaluation. Optimizing prompts individually

for each model could potentially improve their performance, but it would also make comparison less valid because differences in results could come from different prompt quality rather than from actual differences in model capability.

For practical deployment, instance administrators should not assume the prompts used here are optimal. Experimenting with different prompt formulations while monitoring false positive and false negative rates combined with ground truth data would likely improve performance.

5.2.5. Prompt Sensitivity Findings

The prompt sensitivity test showed that models respond very differently to prompt changes. GPT-4o-mini was stable (only 10% of classifications changed), while Llama 3.3 was highly unstable (66.7% changed).

This has several important implications:

Agreement between models may be prompt-dependent. In Chapter 4, we reported 99% agreement between GPT and Llama. However, since Llama changed its decisions for two-thirds of posts when we rephrased the prompt, this high agreement might only work for our specific prompt wording. With different prompts, the agreement could be much lower.

GPT-4o-mini is more reliable for real-world use. Since administrators often need to adapt prompts for their specific platforms, GPT-4o-mini's stability makes it more dependable. Llama needs much more careful prompt design to get consistent results.

The prompt sensitivity test used only 30 posts because of limited resources (testing 4 models $\times$ 4 prompt variants $\times$ 30 posts = 480 API calls). A bigger sample might show different results. Also, we tested only four different prompt versions. There are many possible ways to write prompts, and other versions might give different results. These findings show that the way we write prompts is very important for LLM-based moderation. Our main results work for the specific prompt we used, but users should know that different prompts could give different outcomes. Future research should test more prompt versions to find out which prompt characteristics may lead to more stable results across different models.

5.2.6. Lack of Baseline Comparison

This research compared four LLM architectures against each other but did not evaluate them against simpler baseline methods. Traditional approaches such as keyword matching, rule-based classifiers or traditional machine learning models (such as SVM or Random Forest) might achieve comparable results at significantly lower computational cost. Without this comparison, we cannot determine whether the expense and complexity of LLM-based detection is reasonable. Future work should establish baselines using simpler methods to quantify the added value of LLM analysis.

5.3. Main Results

Our research produced several important findings about using AI to help with content moderation in federated networks.

5.3.1. Technical Implementation

We created a working system with two main parts. The first part uses behavioral filtering to identify potentially suspicious posts based on account characteristics like registration date, number of followers and posting patterns. This filtering reduced the number of posts that need detailed analysis from 175076 to just 566. This makes expensive LLM analysis affordable even for small instances.

The second part analyzes the filtered posts using four different LLMs: GPT-4o-mini, Claude Sonnet 4, Gemini 2.5 Flash Lite and Llama 3.3 70B. We built an interface that shows results from all models in real-time, so moderators can see different perspectives on the same content. This multi-model approach gives moderators more information to make better decisions.

The system connects to Mastodon's streaming API and handles real-time data collection across the federated network using ActivityPub protocol. The architecture is modular, which means instance administrators can adapt it to their specific needs and resources.

5.3.2. Model Agreement Findings

The evaluation showed substantial agreement between the four models. We measured overall agreement using Fleiss' Kappa, which reached 0.756. According to standard interpretation, this represents substantial agreement: much higher than would happen by chance. All four models agreed completely (all marked suspicious or all marked legitimate) on 78.45% of posts. When we looked at pairs of models, we found that three models (GPT-4o-mini, Gemini and Llama) agreed with each other on 98-99% of posts. These three models classified about 61-62% of flagged posts as suspicious. Claude behaved differently, marking 81.80% of posts as suspicious, about 20 percentage points higher than the other three.

This pattern shows that different LLM architectures can reach similar conclusions about content, even though they were trained differently by different companies. The high agreement gives us confidence that automated detection can work consistently. The difference with Claude shows that there are different approaches possible.

An important observation is that the open-source Llama produced results closely aligned with the commercial models. Llama agreed with GPT-4o-mini on 99.29% of posts and demonstrated a nearly identical suspicious-classification rate (61.66% vs. 62.01%). This suggests that, at least within our evaluation framework, Llama's output behavior converges with that of proprietary systems, which might be promising for instances that prefer open-source solutions for reasons of transparency or cost.

The prompt sensitivity analysis showed that models differ significantly in their robustness to prompt variations: GPT-4o-mini demonstrated low sensitivity (10% classification

change rate), while Llama 3.3 showed high sensitivity (66.7%). This finding suggests that the high inter-model agreement observed in the main evaluation may be partly prompt-dependent.

5.4. Practical Implications for Instance Administrators

The results have practical implications for Mastodon instance administrators who want to use automated detection tools.

5.4.1. The System is a Helper, Not a Replacement

> **Core Principle:** All automated classifications function as recommendations to assist human moderators, not as autonomous decisions.

All automated classifications are suggestions that help moderators find potentially problematic content faster. The final decisions about what to do with flagged content must always be made by humans.

There are several reasons why human judgement remains essential:

1. **LLMs make mistakes**. They misunderstand context, miss sarcasm and sometimes flag clearly harmless content. During testing, models occasionally marked normal posts as suspicious.

2. **Content moderation involves value judgments that algorithms cannot make**. What counts as harmful depends on community standards, cultural context and specific situations. For example, a political statement might be acceptable in one case but violate rules in another.

3. **Context matters**. A post that looks suspicious alone might be part of an ongoing legitimate discussion. Moderators who know their community can recognize this context.

4. **Fair moderation requires the ability to explain decisions and consider appeals**. Humans can discuss with users and adjust decisions based on additional information. Automated systems cannot do this (unless deploying AI conversational agents).

The system shows flagged content to moderators with explanations from multiple models. Moderators then review this information and make decisions based on their understanding of community values. The automation handles the repetitive work of scanning large amounts of content, while people make the judgment calls. This approach uses AI for what it does well while keeping humans in control of important decisions.

5.4.2. Practical Deployment Considerations

The research shows that automated detection tools can help Mastodon instances, but success requires understanding both what they can and cannot do. The behavioral filtering achieved its intended purpose in this evaluation: it reduced the workload significantly and the filtered posts showed high model agreement. However, this does not prove the filtering caught all suspicious content. More advanced campaigns using older accounts or avoiding hashtags might get through. Instance administrators should view behavioral filtering as one layer in a multi-layer defense strategy, not as a complete solution. Combining automated detection with user reports, community moderation and regular manual checks provides better coverage than any single approach.

Model selection: GPT-4o-mini, Gemini and Llama provide similar results, so choosing between them depends on other factors. Cost varies significantly: Llama is cheapest ($0.00033-$0.00052 per analysis via Together AI), followed by Gemini ($0.00055-$0.00085), GPT-4o-mini ($0.0008-$0.0013) and Claude ($0.02-$0.03). Llama is the only open-source option, which matters for instances prioritizing transparency or vendor independence. Claude's higher suspicious rate represents a different trade-off: more comprehensive coverage at the cost of more content to review.

Recommended approach: start with behavioral filtering based on account characteristics and hashtag patterns - this alone reduces workload significantly. Then add LLM analysis for filtered posts. Starting with one model is reasonable; multi-model comparison provides additional signal but is not necessary for basic detection. When multiple models agree on a classification, this convergence may be more informative than a single model's output.

Whatever approach an instance chooses, maintaining human oversight is critical. Every flagged post should be reviewed by a person before any action is taken. The automated system helps moderators work more efficiently, but it does not make the final decisions.

5.4.3. Transparency and User Communication

Instances that deploy automated detection should communicate clearly with their communities about how it works. Users should know that:

- Automated systems help moderators by flagging potentially problematic content
- All flagged content is reviewed by human moderators before action is taken
- Specific behavioral patterns (like very new accounts using many hashtags) trigger automated analysis
- Users can appeal if they think their content was unfairly flagged
- The instance takes privacy seriously and only analyzes public posts

Being transparent about automated tools builds trust and helps users understand that technology assists human judgment rather than replacing it.

5.5. Future Research Directions

Several clear next steps would improve on this research.

Ground truth development represents a critical priority for future research. Recruiting domain experts to manually classify a substantial sample of Mastodon posts would enable direct accuracy measurement, revealing whether high model agreement reflects genuine detection capability or systematic biases. Such labeled data would also enable calculation of precision and recall metrics, providing more informative performance assessment than agreement rates alone.

Multilingual evaluation also needs more work. English posts made up 41.7% of the dataset, which means we cannot be sure how well the models work for other languages. Future research should collect more posts in different languages and find people who speak those languages to check if the models work equally well for everyone. This is especially important for instances where users speak many different languages.

Multimodal detection capabilities will grow increasingly important as disinformation evolves. This research focused exclusively on text because current LLMs handle textual content most effectively. However, manipulated images and misleading videos represent major disinformation vectors. As multimodal models improve, extending detection to visual content would increase system comprehensiveness.

Fine-tuning experiments offer potential performance improvements. This evaluation used zero-shot models without task-specific training on Mastodon data. Training models on labeled Mastodon posts might reduce classification errors. This approach is particularly feasible for Llama, where instance administrators could create custom model versions optimized for their specific community's content patterns and moderation priorities.

Finally, evaluating newer model versions would provide updated performance insights. This research used models available during the evaluation period within project budget constraints, but AI companies frequently release improved versions. Regular re-evaluation with state-of-the-art models would track whether disinformation detection capabilities are improving over time.

5.6. Final Remarks

This research began with a question: can volunteer-run Mastodon instances access sophisticated content moderation tools without the massive resources of large tech companies? The findings suggest yes. The technical capability exists: Large Language Models can analyze content with substantial consistency, behavioral filtering makes comprehensive monitoring computationally feasible and open-source models like Llama perform comparably to commercial alternatives. The architecture demonstrated in this work can be deployed by instances with modest technical expertise.

However, the technology is a tool, not a solution. It helps humans work more efficiently but cannot replace human judgment. Content moderation requires understanding context, respecting community values and making difficult decisions about competing goods. These are things that algorithms cannot do, no matter how sophisticated they become.

The substantial agreement between models (Fleiss' Kappa = 0.756) indicates that automated detection can be consistent. The high agreement between GPT-4o-mini, Gemini and Llama shows that different approaches converge on similar assessments. Claude's more conservative approach demonstrates that different thresholds are possible, giving instance administrators choices about how cautious to be.

This thesis demonstrates that AI-assisted moderation can be implemented at the instance level with accessible technology and modest resources. The proof-of-concept system shows that communities can deploy sophisticated detection tools while maintaining autonomy over moderation decisions.

As federated networks grow, they will continue facing challenges from coordinated disinformation and manipulation. The methods developed here provide a technical foundation for addressing these challenges without requiring centralized infrastructure.

The proof-of-concept demonstrates technical feasibility: federated instances can implement AI-assisted content analysis with human moderators retaining final decision authority. Current technology and accessible resources make this approach affordable. Moreover, instances might develop governance frameworks that integrate these tools into human-centered moderation workflows while maintaining transparency and accountability.

6. Conclusion

This thesis investigated the application of Large Language Models to disinformation detection in Mastodon's federated network. The research produced a working proof-of-concept system, a systematic comparison of four LLM architectures and empirical findings about inter-model agreement patterns.

6.1. Summary of Contributions

The research delivered three main contributions. First, we implemented a real-time detection system that combines behavioral filtering with multi-model LLM analysis. The behavioral heuristics reduced the analysis workload from 175076 collected posts to 566 flagged posts, making comprehensive LLM evaluation feasible within modest resource constraints. Second, we conducted a systematic comparison of GPT-4o-mini, Claude Sonnet 4, Gemini 2.5 Flash Lite and Llama 3.3 70B on identical datasets under identical conditions. Third, we demonstrated that the open-source Llama model performed comparably to commercial alternatives. This gives Mastodon servers another option if they want more transparency or if they prefer not to depend on commercial companies.

6.2. Key Findings

The evaluation revealed substantial inter-model agreement with Fleiss' Kappa of 0.756. Three models (GPT-4o-mini, Gemini and Llama) formed a consensus cluster with 98-99% pairwise agreement, classifying approximately 61-62% of flagged posts as suspicious. Claude showed a different pattern, flagging 81.80% of posts as suspicious, about 20 percentage points more than the other models. This difference may be related to its Constitutional AI training approach.

The prompt sensitivity analysis revealed an important issue: Llama changed its classifications for 66.7% of posts when we changed how we wrote the prompts, while GPT-4o-mini only changed its answers for 10% of posts. This result shows that models do not always agree because they understand the content the same way, but sometimes because of how we ask the question. It also shows that some models give more stable results than others.

6.3. Limitations

The biggest limitation of this work is that we do not have ground truth labels: we do not know which posts are actually disinformation and which are not. When models

agree with each other, this shows that they are consistent, but it does not prove they are correct. All models could make the same mistakes because they were trained on similar data. The behavioral filtering works fast and uses less computing power, but advanced disinformation campaigns can avoid it by using older accounts or not using hashtags. Also, most posts in our dataset are in English (41.7%), so we cannot say how well the models work with other languages.

6.4. Future Work

Three improvements could make this research stronger. First, creating datasets where experts label which posts are disinformation would let us measure how accurate the models really are using metrics like precision and recall. Second, testing the system against real disinformation campaigns would show how well it works in practice. Third, studying how human moderators actually use the AI suggestions would help us understand the human side of the system, which is important because people make the final decisions about what content to remove or restrict.

6.5. Closing Remarks

The implementation process provided valuable practical experience with streaming APIs, multi-provider LLM integration and real-time interface design. The iterative process of prompt engineering (understanding why models misinterpret instructions and refining accordingly) offered valuable insights into practical AI development.

This thesis shows that federated social networks can use AI tools to help with moderation without giving up their independence. AI-assisted detection can be set up on individual servers using technology that is available and does not cost too much. This helps volunteer moderators do their work more efficiently while making sure that humans still make the final decisions. The technology is just a tool, not a complete solution, but for small servers that do not have many resources, it can be helpful.

Bibliography

[AAA+23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[act18] Activitypub w3c recommendation. https://www.w3.org/TR/activitypub/, 2018. W3C recommendation.

[AG17] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 2017.

[All24] Bobby Allyn. 2 years in, trump surrogate elon musk has remade x as a conservative megaphone. https://www.npr.org/2024/10/22/nx-s1-5156184/elon-musk-trump-election-x-twitter, October 2024.

[ATS17] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. In *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments (ISDDC 2017)*. Springer, 2017.

[BHA+21] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[BHG+12] Adam Bielenberg, Leif Helm, Anthony Gentilucci, Dan Stefanescu, and Honggang Zhang. The growth of diaspora – a decentralized online social network in the wild. In *Proceedings of IEEE INFOCOM Workshops*, 2012.

[BKK+22] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

[Cas10] Bettina J. Casad. Confirmation bias. https://www.britannica.com/science/confirmation-bias, 2010. Encyclopedia article.

[FKKG24] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 2024.

[Fle71] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[FVD+16] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 2016.

[GDS+24] Aryo Pradipta Gema, Luke Daines, Pasquale Sivakumar, Aryaz Arabnejad, and Hao Wu. Parameter-efficient fine-tuning of LLaMA for the clinical domain. *arXiv preprint arXiv:2307.03042*, 2024.

[Gem23a] Gemini Team. Gemini: A family of highly capable multimodal models. Technical report, Google, 2023.

[Gem23b] Gemini Team, Google. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[GSSZ] Dan Grippi, Maxwell Salzberg, Raphael Sofaer, and Ilya Zhitomirskiy. Diaspora is a nonprofit, user-owned, distributed social network. https://en.wikipedia.org/wiki/Diaspora_(social_network).

[HV22] Kristina Hook and Ernesto Verdeja. Social media misinformation and the prevention of political instability and mass atrocities. Technical report, Stimson Center, 2022.

[Iea20] Md Saiful Islam and et al. Covid-19–related infodemic and its impact on public health: A global social media analysis. *American Journal of Tropical Medicine and Hygiene*, 2020.

[KBD24] Deepak Kumar, Yousef AbuHashem Bloomfield, and Zakir Durumeric. Watch your language: Investigating content moderation with large language models. In *Proceedings of the International AAAI Conference on Web and Social Media*, 2024.

[KGH14] Adam D. I. Kramer, Jamie E. Guillory, and Jeffrey T. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 111(24):8788–8790, June 2014.

[KRDE25] Shanu Kumar, Gauri Rani, Saurabh Dandapat, and Asif Ekbal. Socio-culturally aware evaluation framework for LLM-based content moderation. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025.

[LBB+18] David M. J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. The science of fake news. *Science*, 2018.

[LCGT21] Lucio La Cava, Sergio Greco, and Andrea Tagarelli. Understanding the growth of the fediverse through the lens of mastodon. In *Applied Network Science*, 2021.

[LFAH24] Lingyao Li, Lizhou Fan, Shubham Atreja, and Libby Hemphill. "hot" chatgpt: The promise of chatgpt in detecting and discriminating hateful, offensive, and toxic comments on social media. *ACM Transactions on the Web*, 2024.

[Mas25] Mastodon gGmbH. Mastodon – about. `https://mastodon.social/about`, 2025.

[MH25] Filippo Menczer and Thomas T. Hills. AI-driven disinformation: policy recommendations for democratic resilience. *Humanities and Social Sciences Communications*, 2025.

[RAL+25] Kristina Radivojevic, D. J. Adams, Griffin Laszlo, Felixander Kery, and Tim Weninger. User migration in the twitter diaspora. *EPJ Data Science*, 14:36, 2025.

[RHMS23] Sarthak Roy, Ashish Harshvardhan, Animesh Mukherjee, and Punyajoy Saha. Probing llms for hate speech detection: strengths and vulnerabilities. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, 2023.

[RJDC+19] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Challenges in the decentralised web: The mastodon case. In *Proceedings of the Internet Measurement Conference*, 2019.

[SUS25] Isaac Slaughter, Johan Ugander, and Martin Saveski. Community notes reduce the spread of misinformation on social media. *Proceedings of the National Academy of Sciences*, 2025.

[TDKM23] Shalini Talwar, Amandeep Dhir, Puneet Kaur, and Marcello Mariani. Motives for posting fake reviews: Evidence from a cross-cultural comparison. *Journal of Business Research*, 2023.

[TJLL18] Edson C Tandoc Jr, Zheng Wei Lim, and Richard Ling. Defining "fake news": A typology of scholarly definitions. *Digital journalism*, 2018.

[TMS+23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti

Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint*, abs/2307.09288, 7 2023. Preprint — as of July 2023.

[UBC22] Joshua Uyheng, Daniele Bellutta, and Kathleen M. Carley. Bots amplify and redirect hate speech in online discourse about racism during the covid-19 pandemic. *Social Media + Society*, 2022.

[Vea17] Ashish Vaswani and et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[VFD+17] Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the International AAAI Conference on Web and Social Media*, 2017.

[VRA18] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 2018.

[Wika] Wikipedia. Fleiss's kappa. `https://en.wikipedia.org/wiki/Fleiss%27s_kappa`.

[Wikb] Wikipedia contributors. Bluesky. `https://en.wikipedia.org/wiki/Bluesky`.

[Wikc] Wikipedia contributors. Bluesky – growth and user base. `https://en.wikipedia.org/wiki/Bluesky`.

[YVD+19] Kai-Cheng Yang, Onur Varol, Clayton A. Davis, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies*, 2019.

[Zar20] John Zarocostas. How to fight an infodemic. *The Lancet*, 2020.

[ZGR18] Matteo Zignani, Sabrina Gaito, and Gian Paolo Rossi. Follow the "mastodon": Structure and evolution of a decentralized online social network. In *Proceedings of the Twelfth International AAAI Conference on Web and Social Media (ICWSM 2018)*, pages 541–551, 2018.

[ZZ20] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys*, 2020.

A. Appendix

A.1. Sample Model Outputs

Below are examples of model responses for the same posts. Each example shows the JSON output from different AI models analyzing suspicious content.

A.1.1. Example 1

OpenAI Response


```
{
    "is_suspicious": true,
    "confidence": 0.8,
    "reasoning": "The post contains a reference to a film by Abbas
    Kiarostami, which could indicate a genuine interest in cinema. However,
    the lack of hashtags, the author's very low follower count (1 follower
    and 0 followings), and the fact that the account was created just 24
    days before the post raises concerns about the authenticity of the
    account. Additionally, the author has only made 8 posts, which suggests
    low engagement.",
    "red_flags": [
        "Low follower count (1 follower)",
        "No followings (0 followings)",
        "Recent account creation (24 days before post)",
        "Limited posting history (only 8 posts)"
    ]
}
```


Claude Response


```
{
    "is_suspicious": true,
    "confidence": 0.42,
    "reasoning": "The post exhibits several concerning patterns that suggest
    potential suspicious activity. The account was created very recently
    (November 3, 2025) and has an unusual follower profile with only 1
    follower and 0 followings, which is atypical for organic social media
    behavior. The posting frequency is relatively high for such a new
    account (8 posts in 24 days). The content itself appears to reference a
```

```
    film by Abbas Kiarostami, but there's a discrepancy - the post claims
    the film is from 1970, while Kiarostami's notable works began later in
    his career. The post lacks hashtags and has minimal engagement
    indicators. However, the specific cultural reference and non-English
    text (appears to be Catalan) suggest some level of human curation,
    creating ambiguity in the assessment.",
    "red_flags": [
        "Very new account (24 days old)",
        "Unusual follower ratio (1 follower, 0 following)",
        "High posting frequency for new account",
        "Potential factual inaccuracy in film date",
        "Minimal text with external link pattern",
        "No hashtags despite film/cultural content",
        "Account behavior inconsistent with typical user engagement"
    ]
}
```


Gemini Response


```
{
    "is_suspicious": true,
    "confidence": 0.70,
    "reasoning": "The post itself is a simple share of a film title and a
    YouTube link, which is not inherently suspicious. However, the author's
    profile exhibits several red flags. The account was registered very
    recently (November 3, 2025) and has a minimal posting history (8 posts).
    Crucially, the author has 1 follower and follows 0 accounts, indicating
    a lack of genuine social interaction or established presence. While the
    content is not overtly AI-generated, the combination of a brand new
    account with no social connections and a very limited posting history,
    coupled with the immediate posting activity, raises suspicion. It could
    be an automated account for spamming links, or a newly created account
    for a specific, potentially manipulative purpose.",
    "red_flags": [
        "Newly created account with minimal activity",
        "Zero followings and very few followers",
        "High posting frequency relative to account age and social
        engagement",
        "Lack of established social presence"
    ]
}
```

A.1.2. Example 2

OpenAI Response


```
{
    "is_suspicious": true,
    "confidence": 0.8,
    "reasoning": "The post exhibits characteristics typical of AI-generated
    content, such as a structured and overly formal tone that lacks personal
    touch or engagement. The author has a very low follower count (0) and
    has only been active since November 11, 2025, which is a short time
    before posting. The post includes a link to an external site, which is
    common in spammy or promotional posts. Additionally, the hashtags used
    are generic and overly promotional, indicating a potential attempt to
    reach a broad audience without genuine engagement.",
    "red_flags": [
        "Author has 0 followers and 0 followings, indicating a new or
        inactive account.",
        "Post contains a promotional link, which is often associated with
        spam.",
        "Hashtags are generic and lack specificity, suggesting a lack of
        genuine content.",
        "The account was created just 16 days before the post, indicating a
        possible lack of authenticity."
    ]
}
```


Claude Response


```
{
    "is_suspicious": true,
    "confidence": 0.42,
    "reasoning": "The post exhibits several concerning patterns that suggest
    suspicious activity, though some factors create ambiguity. The account
    shows classic spam characteristics: newly created (16 days old), zero
    followers/following, and promotional content with commercial links. The
    text structure appears formulaic and promotional rather than organic
    user-generated content. However, the relatively low post count (10
    posts) and coherent, grammatically correct content prevent higher
    confidence in the assessment. The hashtag usage, while extensive, is
    topically relevant rather than completely random spam.",
    "red_flags": [
        "New account created 16 days ago with immediate promotional posting",
        "Zero followers and zero following indicates no genuine social
        engagement",
```

```
        "Commercial link promotion for visa services",
        "Formulaic, marketing-style language structure",
        "High hashtag density typical of promotional content",
        "Content reads like marketing copy rather than personal experience",
        "No punctuation between main text and link creates awkward
        formatting"
    ]
}
```


Llama Response


```
{
    "is_suspicious": true,
    "confidence": 0.5,
    "reasoning": "The post exhibits several characteristics that raise
    suspicions of being AI-generated or spammy. Firstly, the author has
    zero followers and followings, yet they have already posted ten times
    since their registration on 2025-11-11. This suggests an unusual posting
    frequency without any social engagement, which is a common trait of
    automated or spam accounts. Additionally, the post itself contains a
    very specific and promotional link, which could indicate affiliate
    marketing or spam. The hashtags, although relevant to travel and
    tourism, are numerous and might be intended to reach a broad audience in
    a way that's more typical of spam posts.",
    "red_flags": [
        "New account with high posting frequency",
        "Zero social engagement (followers and followings)",
        "Promotional link in the post",
        "Excessive use of relevant hashtags",
        "Short time between account creation and active posting"
    ]
}
```


A.1.3. Example 3

OpenAI Response


```
{
    "is_suspicious": true,
    "confidence": 0.75,
    "reasoning": "The post contains a long string of hashtags that appear to
    be excessive and not entirely relevant to the content, which is a common
    characteristic of spammy or AI-generated posts. The author has a very
    low follower count (20) and a low number of followings (9), suggesting
```

```
    they may not be an active or genuine user. Additionally, the account was
    created less than a month before the post was made, indicating a rapid
    posting behavior that could be indicative of suspicious activity. The
    overall structure of the post, with a lack of coherent narrative and an
    overabundance of hashtags, further supports the assessment of being
    AI-generated or spammy.",
    "red_flags": [
        "Excessive and irrelevant hashtags",
        "Low follower count and followings",
        "Recent account creation relative to posting activity",
        "Lack of coherent narrative in the post"
    ]
}
```


Claude Response


```
{
    "is_suspicious": true,
    "confidence": 0.55,
    "reasoning": "The post exhibits several concerning patterns that suggest
    potential AI generation or suspicious activity. The account is very new
    (registered 23 days ago) but has already made 21 posts, indicating high
    posting frequency. The post text shows clear formatting issues with
    missing spaces between elements ('Cow#Oil', 'cmwww.jurakuba.com#cow'),
    which is characteristic of automated content generation. The excessive
    use of hashtags (30+ tags) covering multiple languages (English and
    German) appears spammy and designed for maximum reach rather than
    genuine engagement. However, the content itself is coherent and relates
    to a specific artwork, and the website reference suggests potential
    legitimate artistic promotion, which prevents higher confidence in the
    suspicious assessment.",
    "red_flags": [
        "Very new account (23 days old)",
        "High posting frequency (21 posts in 23 days)",
        "Formatting errors and missing spaces in text",
        "Excessive hashtag usage (30+ hashtags)",
        "Multi-language hashtag spam pattern",
        "Potential automated content generation indicators"
    ]
}
```


Gemini Response


```
{
```

```
    "is_suspicious": true,
    "confidence": 0.70,
    "reasoning": "The post exhibits several characteristics that suggest it
    might be AI-generated or part of a spam campaign. The extremely high
    density of highly specific and somewhat redundant hashtags (e.g., 'cow',
    'cows', 'calf', 'farm', 'farmanimals', 'animal') is a common tactic used
    by bots or spammers to maximize visibility, often without genuine
    engagement. The author's profile is also a significant red flag: a very
    recent registration date (November 4, 2025) with a low number of posts
    (21) and a very low follower-to-following ratio (20 followers, 9
    followings) suggests a newly created account likely used for automated
    posting. The post text itself is a simple description of an artwork with
    a URL, which is typical for promotional spam.",
    "red_flags": [
        "Excessive and redundant hashtags",
        "Newly registered account",
        "Low follower count relative to followings",
        "Potentially automated posting pattern",
        "Promotional link in post"
    ]
}
```


A.1.4. Example 3

OpenAI Response


```
{
    "is_suspicious": true,
    "confidence": 0.75,
    "reasoning": "The post promotes a digital detox, which is a common theme
    in wellness discussions, but the phrasing and structure are somewhat
    generic and could be indicative of AI-generated content. The author has
    only 1 follower and follows 19 accounts, which suggests a lack of
    engagement or a newly created account. The account was registered just
    14 days before the post was made, indicating a rapid entry into content
    creation without a substantial history. The hashtags used are relevant
    to the content but could also be seen as an attempt to gain visibility
    without genuine engagement.",
    "red_flags": [
        "Newly created account with low follower count",
        "Generic phrasing typical of AI-generated content",
        "Rapid posting activity after account creation"
    ]
}
```

Claude Response


```
{
    "is_suspicious": true,
    "confidence": 0.58,
    "reasoning": "This post exhibits several characteristics commonly
    associated with AI-generated or promotional content. The account was
    created very recently (November 13, 2025) and has already made 33 posts
    in just 14 days, indicating high-frequency posting typical of automated
    accounts. The follower-to-following ratio (1:19) suggests minimal
    organic engagement. The post text follows a formulaic structure with
    motivational language, emoji usage, and promotional hashtags that are
    characteristic of AI-generated wellness content. The content promotes a
    specific blog post with tracking parameters, indicating commercial
    intent.",
    "red_flags": [
        "Very new account (15 days old)",
        "High posting frequency (33 posts in 14 days)",
        "Poor follower ratio (1 follower, 19 following)",
        "Formulaic motivational content structure",
        "Commercial link with tracking parameters",
        "Hashtag optimization pattern",
        "Generic wellness/self-help messaging",
        "Missing space after quotation mark in text"
    ]
}
```


Gemini Response


```
{
    "is_suspicious": true,
    "confidence": 0.85,
    "reasoning": "The post exhibits several indicators of being potentially
    AI-generated or part of a coordinated, low-effort campaign. The author's
    account is very new (registered just 14 days prior to the post) and has
    a minimal follower count (1 follower) with a slightly higher following
    count (19), suggesting a lack of organic engagement. The post text
    itself is generic and uses common, slightly overly enthusiastic phrasing
    (’’, ’’, ’truly matters’) often seen in AI-generated content designed
    for broad appeal. The inclusion of a UTM parameter in the link suggests
    a deliberate tracking mechanism, which can be employed by both
    legitimate marketers and potentially by automated systems pushing
    content.",
    "red_flags": [
```

```
        "Very new account registration date (2025-11-13)",
        "Extremely low follower count (1)",
        "Generic and slightly overly enthusiastic post text",
        "Presence of UTM parameters in the shared link, suggesting tracking
        or automated promotion",
        "Content topic (digital detox) is a common theme for easily
        generated or promoted content"
    ]
}
```

A.2. Prompt Variants for Sensitivity Testing

This appendix shows the four different prompt versions that we used in the prompt sensitivity tests. Each version was designed to check how different ways of writing prompts affect the model's decisions and confidence scores.

A.2.1. Baseline Prompt

The baseline prompt represents the standard version used in the main analysis. It includes complete guidelines and a detailed confidence scale.

```
You are an expert in social media content analysis with
   specialized knowledge in detecting AI-generated or suspicious
   posts.
Your task is to analyze the following Mastodon post and evaluate
   whether it is likely suspicious or AI-generated.

**Current Date and Time:** {current_date}

**Post Analysis Details:**

1. **Post Text:**
   "{content}"

2. **Author Details:**
   - Number of followers: {author_followers_count}
   - Number of followings: {author_following_count}
   - Total posts written by the author: {author_statuses_count}
   - Date of registration: {author_created_at}

3. **Post Metadata:**
   - Post created at: {created_at}
   - Hashtags used in the post: check hashtags in the post text

**Guidelines for Evaluation:**
- Consider whether the post text contains patterns commonly found
   in AI-generated content
```

```
- Assess the author's activity profile (registration date, posting
    frequency, follower ratio)
- Examine the hashtags: Are they relevant or spammy?
- Consider the time between account creation and post creation
- Take into account the current date when evaluating temporal
   aspects
- Combine all factors to determine the likelihood of being
   suspicious

**Response Format (JSON ONLY):**
{
    "is_suspicious": true/false,
    "confidence": 0.0-1.0,
    "reasoning": "detailed explanation based on the parameters",
    "red_flags": ["flag1", "flag2", ...]
}

Confidence scale:
- 0.9-1.0: Very certain
- 0.7-0.89: Confident
- 0.5-0.69: Moderately confident
- 0.3-0.49: Uncertain
- 0.0-0.29: Very uncertain
```

A.2.2. Simplified Prompt

The simplified variant removes detailed guidelines and explanation text to test whether shorter instructions affect how well the classification works.

```
Analyze this Mastodon post and determine if it appears suspicious
   or AI-generated.

Current Date: {current_date}

Post Text: "{content}"

Author Info:
- Followers: {author_followers_count}
- Following: {author_following_count}
- Total posts: {author_statuses_count}
- Registered: {author_created_at}

Respond with JSON only:
{
    "is_suspicious": true/false,
    "confidence": 0.0-1.0,
    "reasoning": "your explanation",
    "red_flags": ["flag1", "flag2", ...]
}
```

A.2.3. Moderator Prompt

The moderator variant changes the task description to content moderation instead of AI detection, testing whether how we describe the task affects the classification decisions.

```
You are a content moderator for a social media platform. Your job is to review flagged posts and decide if they violate community guidelines or appear to be spam/bot activity.

Current Date: {current_date}

**Flagged Post:**
"{content}"

**Account Information:**
- Followers: {author_followers_count}
- Following: {author_following_count}
- Total posts: {author_statuses_count}
- Account created: {author_created_at}

Review this post and determine if it should be escalated for further review.

**Response Format (JSON ONLY):**
{
    "is_suspicious": true/false,
    "confidence": 0.0-1.0,
    "reasoning": "your moderation decision rationale",
    "red_flags": ["flag1", "flag2", ...]
}

Confidence scale:
- 0.9-1.0: Very certain
- 0.7-0.89: Confident
- 0.5-0.69: Moderately confident
- 0.3-0.49: Uncertain
- 0.0-0.29: Very uncertain
```

A.2.4. No-Scale Prompt

The no-scale variant is identical to the baseline but without the confidence scale definition at the end. This tests whether explicit calibration guidance affects confidence scores.

```
You are an expert in social media content analysis with specialized knowledge in detecting AI-generated or suspicious posts.
Your task is to analyze the following Mastodon post and evaluate whether it is likely suspicious or AI-generated.

**Current Date and Time:** {current_date}
```

```
5
6  **Post Analysis Details:**
7
8  1. **Post Text:**
9     "{content}"
10
11 2. **Author Details:**
12    - Number of followers: {author_followers_count}
13    - Number of followings: {author_following_count}
14    - Total posts written by the author: {author_statuses_count}
15    - Date of registration: {author_created_at}
16
17 3. **Post Metadata:**
18    - Post created at: {created_at}
19    - Hashtags used in the post: check hashtags in the post text
20
21 **Guidelines for Evaluation:**
22 - Consider whether the post text contains patterns commonly found
      in AI-generated content
23 - Assess the author's activity profile (registration date, posting
       frequency, follower ratio)
24 - Examine the hashtags: Are they relevant or spammy?
25 - Consider the time between account creation and post creation
26 - Take into account the current date when evaluating temporal
      aspects
27 - Combine all factors to determine the likelihood of being
      suspicious
28
29 **Response Format (JSON ONLY):**
30 {
31     "is_suspicious": true/false,
32     "confidence": 0.0-1.0,
33     "reasoning": "detailed explanation based on the parameters",
34     "red_flags": ["flag1", "flag2", ...]
35 }
```

B. Database Schema

This appendix contains the complete database schema.

B.1. Tables Overview

The database consists of seven main tables:

- `instances` – Fediverse instance metadata
- `accounts` – User account information
- `raw_statuses` – Raw status data before processing
- `statuses` – Processed status data
- `statuses_to_check` – Statuses flagged for AI analysis
- `trends` – Trending hashtags and topics
- `suspicious_trends` – Trends identified as potentially suspicious

B.2. Table Definitions

B.2.1. instances

Stores metadata about Fediverse instances.

```
CREATE TABLE instances (
    id                  VARCHAR(24) NOT NULL PRIMARY KEY,
    name                VARCHAR(50),
    added_at            TIMESTAMP,
    updated_at          TIMESTAMP,
    checked_at          TIMESTAMP,
    uptime              INTEGER,
    up                  BOOLEAN,
    dead                BOOLEAN,
    version             VARCHAR,
    ipv6                BOOLEAN,
    https_score         INTEGER,
    https_rank          VARCHAR,
```

```
    obs_score          INTEGER,
    obs_rank           VARCHAR,
    users              VARCHAR,
    statuses           VARCHAR,
    connections        VARCHAR,
    open_registrations BOOLEAN,
    info               JSON,
    thumbnail          VARCHAR,
    thumbnail_proxy    VARCHAR,
    active_users       INTEGER,
    email              VARCHAR,
    admin              VARCHAR
);
```

B.2.2. accounts

Stores user account information from Fediverse instances.

```
CREATE TABLE accounts (
    id              VARCHAR NOT NULL PRIMARY KEY,
    username        VARCHAR,
    acct            VARCHAR,
    display_name    VARCHAR,
    locked          BOOLEAN,
    bot             BOOLEAN,
    discoverable    BOOLEAN,
    "group"         BOOLEAN,
    created_at      TIMESTAMP,
    note            VARCHAR,
    url             VARCHAR,
    avatar          VARCHAR,
    avatar_static   VARCHAR,
    header          VARCHAR,
    header_static   VARCHAR,
    followers_count INTEGER,
    following_count INTEGER,
    statuses_count  INTEGER,
    last_status_at  TIMESTAMP,
    instance_url    VARCHAR
);

CREATE UNIQUE INDEX accounts_acct_index
    ON accounts (acct);
CREATE INDEX accounts_instance_url_index
```

```
    ON accounts (instance_url);
```

B.2.3. raw_statuses

Stores unprocessed status data as received from instances.

```
CREATE TABLE raw_statuses (
    id                      VARCHAR NOT NULL PRIMARY KEY,
    created_at              TIMESTAMP,
    in_reply_to_id          VARCHAR,
    in_reply_to_account_id VARCHAR,
    sensitive               BOOLEAN,
    spoiler_text            VARCHAR,
    visibility              VARCHAR,
    language                VARCHAR,
    uri                     VARCHAR,
    url                     VARCHAR,
    replies_count           INTEGER,
    reblogs_count           INTEGER,
    favourites_count        INTEGER,
    quotes_count            INTEGER,
    quote                   VARCHAR,
    quote_approval          VARCHAR,
    edited_at               TIMESTAMP,
    content                 VARCHAR,
    tags                    VARCHAR[]
);
```

B.2.4. statuses

Stores processed status data with normalized identifiers.

```
CREATE TABLE statuses (
    id                      VARCHAR NOT NULL PRIMARY KEY,
    created_at              TIMESTAMP,
    in_reply_to_id          BIGINT,
    in_reply_to_account_id BIGINT,
    sensitive               BOOLEAN,
    spoiler_text            VARCHAR,
    visibility              VARCHAR,
    language                VARCHAR,
    uri                     VARCHAR,
    url                     VARCHAR,
    replies_count           INTEGER,
```

```
    reblogs_count           INTEGER,
    favourites_count        INTEGER,
    quotes_count            INTEGER,
    quote                   VARCHAR,
    quote_approval          VARCHAR,
    edited_at               TIMESTAMP,
    content                 VARCHAR,
    tags                    VARCHAR[]
);

CREATE INDEX statuses_tags_index
    ON statuses USING GIN (tags);
```

B.2.5. statuses_to_check

Stores statuses flagged for analysis with AI model responses.

```
CREATE TABLE statuses_to_check (
    id                      VARCHAR NOT NULL PRIMARY KEY,
    created_at              TIMESTAMP NOT NULL,
    language                VARCHAR,
    url                     VARCHAR,
    content                 VARCHAR,
    is_suspicious           BOOLEAN,
    chatgpt_response        VARCHAR,
    openai_response         TEXT,
    openai_confidence       DOUBLE PRECISION,
    openai_is_suspicious    BOOLEAN,
    claude_response         TEXT,
    claude_confidence       DOUBLE PRECISION,
    claude_is_suspicious    BOOLEAN,
    gemini_response         TEXT,
    gemini_confidence       DOUBLE PRECISION,
    gemini_is_suspicious    BOOLEAN,
    llama_response          TEXT,
    llama_confidence        DOUBLE PRECISION,
    llama_is_suspicious     BOOLEAN,
    checked_at              TIMESTAMP,
    author_followers_count  INTEGER,
    author_following_count  INTEGER,
    author_statuses_count   INTEGER,
    author_created_at       TIMESTAMP
);
```

B.2.6. trends

Stores trending hashtags and topics across instances.

```
CREATE TABLE trends (
    id                      SERIAL PRIMARY KEY,
    name                    VARCHAR,
    url                     VARCHAR,
    uses_in_last_seven_days INTEGER
);
```

B.2.7. suspicious_trends

Stores trends identified as potentially containing suspicious content.

```
CREATE TABLE suspicious_trends (
    id                          BIGSERIAL PRIMARY KEY,
    name                        VARCHAR,
    url                         VARCHAR NOT NULL,
    uses_in_last_seven_days     INTEGER,
    number_of_accounts          INTEGER,
    instance_url                VARCHAR,
    number_of_similar_statuses INTEGER DEFAULT 0
);

CREATE UNIQUE INDEX suspicious_trends_url_index
    ON suspicious_trends (url);
CREATE INDEX suspicious_trends_instance_url_index
    ON suspicious_trends (instance_url);
```