Deconstructing the AI Constitution Winter 2025

Marek Mytkowski[*]
01171322@pw.edu.pl

Weronika Gozdera[*]
01171293@pw.edu.pl

Julia Dudzińska[*]
01171286@pw.edu.pl

Wojciech Michaluk[*]
01171253@pw.edu.pl

supervisor: Anna Wróblewska[*]
anna.wroblewska1@pw.edu.pl

initiator of the study: Bartosz Pieliński[†]
b.pielinski@uw.edu.pl

Abstract

System prompts define the high-level behaviour, constraints, and identity of large language models, yet their structure and evolution remain insufficiently studied. In this work, we analyse leaked system prompts from major LLM providers using a simplified Institutional Grammar (IG) framework. We decompose prompts into constitutive statements, which define model identity and scope, and regulative statements, which govern permitted, required, or prohibited actions.

Using an LLM-assisted, expert-validated annotation pipeline, we study how system prompts change over time and differ across providers. Our analysis focuses on prompt length, the proportion of non-code instructional content, textual readability, the distribution of deontic types in regulative statements and semantic provider-specific clustering via Sentence-BERT embeddings and hierarchical analysis. The results show a consistent growth in prompt size over time, a relative decline of non-code content, and systematic differences between providers in both prompt complexity and normative style. These findings provide empirical insight into emerging design patterns in LLM system prompts and demonstrate the usefulness of IG-based analysis for studying AI governance artefacts.

[*] Warsaw University of Technology
[†] University of Warsaw

1 Introduction

System prompts play a central role in governing the behaviour of large language models (LLMs). They define high-level instructions, constraints, and self-descriptions that shape how a model responds to user inputs, interacts with tools, and adheres to safety or policy requirements. Despite their importance, system prompts are rarely documented publicly and lack standardised methods for analysis and comparison across models and providers.

System prompts have received relatively limited attention in existing research, particularly with respect to their internal structure and syntactic composition. This gap is largely due to the lack of transparency of proprietary models and the limited availability and reliability of data sources. Instead, most prior work has focused on *prompting* more broadly, rather than on system prompts as institutional artefacts.

First, a substantial body of research addresses how to design effective end-user prompts. This includes works that propose general prompting guidelines [2], methods for teaching non-experts how to write prompts [9], and analyses of why non-experts often fail to prompt effectively [19]. While some of these insights may generalise to system prompts, this line of work does not explicitly study prompt structure or syntax.

Second, several studies investigate prompt reconstruction or extraction indirectly, for example, by reverse engineering prompts from model outputs or internal representations [20]. Other work examines the effects of prompting strategies on learning outcomes, particularly in higher education settings [11]. These studies focus on the downstream effects of prompts rather than their internal organisation.

Third, a smaller set of works comes closer to the present study by analysing system-level or role-based prompts. This includes research on how the position of a demographically identifiable prompt (system versus user) affects bias in model responses [12], how system prompts shape model identity [13], and how specifying a persona influences factual accuracy [22]. While methodologically related, these studies do not analyse system prompts as structured collections of institutional statements, nor do they compare prompt design trends across providers.

To the best of our knowledge, there is no prior work that systematically analyses leaked system prompts of proprietary LLMs, regardless of analytical scope. In this sense, our study represents an initial attempt to examine the structure, evolution, and cross-provider differences of real-world system prompts using a formal institutional framework.

In this project, we developed and applied an analytical framework to systematically and comparatively study LLM system prompts. We focus on leaked system prompts collected from the CL4R1T4S repository and analyse them using Institutional Grammar (IG). IG allows us to distinguish between *constitutive statements*, which define what the system is (e.g., its identity, role, or capabilities), and *regulative statements*, which define what the system is allowed, required, or forbidden to do.

A key challenge addressed in this work is the lack of established preprocessing and annotation procedures suitable for system prompts. Unlike legal or policy texts, system prompts are often informal, partially implicit, and not always grammatically well-formed. To address this, we introduce a simplified IG schema and an LLM-accelerated annotation pipeline with expert validation, enabling consistent decomposition of prompts into atomic statements and IG components.

Using this framework, we analyse prompt evolution and provider-specific differences along several dimensions. First, we study how prompt length and structure change over time, using character count, the Non-Code-to-Whole (NCtW) ratio and the Constitutive-Statements-to-All (CStA) ratio. Second, we examine textual complexity using the Flesch Reading Ease score to compare readability across providers. Third, we analyse regulative statements by deontic type (command, permission, prohibition, strategy) to characterise normative styles embedded in system prompts. Fourth, we apply Sentence-BERT embeddings to construct a semantic "prompt-space" mapping, using Spectral clustering, UMAP visualisation, and hierarchical dendrograms to identify provider-specific semantic signatures and evolutionary relationships.

The study is guided by the following research questions:

RQ1 What preprocessing and annotation steps are required to reliably decompose system prompts into constitutive and regulative statements using Institutional Grammar?

RQ2 Are there general trends in the evolution of system prompt size, structure, and complexity over time?

RQ3 Do LLM providers differ systematically in how they formulate system prompts, particularly with respect to readability and normative (deontic) structure?

By answering these questions, this work provides an empirically grounded view of how system prompts are designed and evolve. More broadly, it demonstrates that IG provides a useful analytical lens for studying system prompts as institutional artefacts that encode governance, control, and design decisions in AI systems.

2 Institutional Grammar 2.0 as an Analytical Framework

Institutional Grammar (IG) is a formal framework for analysing institutions through the linguistic structure of statements that define, constrain, or enable behaviour. It was originally introduced by Crawford and Ostrom in 1995 as a method for decomposing institutional statements into a small set of analytically meaningful components [4]. In its initial formulation (IG 1.0), the framework focused on five syntactic elements: *Attributes* (the actor), *Deontic* (the prescriptive operator), *Aim* (the action), *Conditions* (the context under which the action applies), and *Or else* (sanctions) [4, 7].

Subsequent work by Frantz, Siddiki, and others extended this approach into Institutional Grammar 2.0 (IG 2.0), which shifts the framework from a purely syntactic decomposition toward a more explicitly semantic representation of institutional meaning [6, 7]. A key extension introduced in

IG 2.0 is the formal treatment of *constitutive statements*. These statements do not regulate behaviour directly, but instead define entities, roles, artefacts, and statuses that parameterise the institutional system. IG 2.0 also supports nesting and decomposition of complex directives into atomic units, enabling systematic analysis of multi-part and hierarchically structured instructions.

Within the IG framework, regulative statements can be further differentiated into *strategies*, *norms*, and *rules*, depending on their degree of prescriptiveness [4, 7]. Strategies describe regularised behaviour without explicit normative force and typically lack a deontic component. Norms introduce a deontic element, such as permission or obligation, but do not specify explicit sanctions. Rules represent the strongest form of regulation, combining a deontic component with an explicit consequence for non-compliance. This distinction provides a useful lens for analysing the strength and formality of behavioural constraints expressed in system prompts, even when sanctions are only implicitly stated.

This framework is well-suited for analysing the system prompts of large language models. System prompts serve as internal governance mechanisms that define both the model's identity and the constraints under which it operates. They establish a clear hierarchy in which developer-provided instructions take precedence over user inputs, and they often encode strong normative expectations using imperative language, constraints, and conditional rules. From an institutional perspective, system prompts can therefore be understood as collections of constitutive and regulative statements that jointly specify the model's role, capabilities, and permitted actions.

Applying IG 2.0 to system prompts makes it possible to distinguish between statements that define what a model *is*—for example, its persona, scope, or intended function—and statements that define what the model *should, may, or must not do*. They often combine natural-language instructions with technical elements such as formatting rules, schemas, or tool specifications. IG provides a principled way to parse these heterogeneous texts into comparable components and to make implicit governance structures explicit.

At the same time, system prompts differ substantially from the legal and policy texts for which IG was originally developed. They are often informal, partially implicit, and not always grammatically well-formed. As a result, direct application of the full IG 2.0 specification is not always feasible. In this work, IG is therefore used as an analytical lens rather than a rigid formalism. Core conceptual distinctions—most importantly between constitutive and regulative statements—are preserved, while the concrete set of components is adapted to the empirical properties of the data. The specific simplifications and annotation rules adopted in this study are described in the Section 5.

3 Security Risks of Leaked System Prompts

System prompts in LLMs serve as hidden instructions that govern model behaviour, yet their leaked nature raises important limitations for any empirical analysis. Unlike officially released specifications, leaked system prompts typically lack verifiable provenance. It is often unclear whether a given prompt corresponds to a production system, an experimental configuration, or a partially reconstructed artefact obtained through reverse engineering. From an analytical standpoint, this uncertainty introduces the risk of drawing conclusions from texts that may be incomplete, outdated, or syntactically altered, thereby distorting observed structural patterns and biasing comparative results.

These limitations are particularly relevant for syntactic and structural analyses, where even small modifications—such as reordered clauses, removed sections, or paraphrased instructions—can affect statement segmentation, classification, and quantitative measures. If not accounted for, such noise may lead to overestimating or underestimating similarities between prompts, misidentifying provider-specific conventions, or attributing structural trends to design choices that are in fact artefacts of data leakage and reconstruction. For this reason, the use of leaked system prompts poses an inherent methodological risk: conclusions must be framed as probabilistic and contingent on data reliability. This motivates the additional validation steps adopted in this study.

In the referenced guide [14], the authors analyse a public repository of leaked system prompts—the same repository [3] we have used in our project. Because security researchers and professionals already rely on that repository, its use supports the authenticity and relevance of our analysis.

Additionally, to confirm the reliability of the selected repository, we decided to verify its similarity to other datasets of this type. We assumed that the occurrence of similar or identical system prompts across multiple repositories, independent of each other, indicates their authenticity and, given the lack of further verification, is a strong indication that the dataset under study is sufficiently reliable.

We selected four publicly available repositories as comparative datasets, hereinafter referred to as reference datasets (in distinction from the candidate dataset – CL4R1T4S): Awesome AI System Prompts (AASP) [1], System Prompts and Model of AI Tools (SPaMoAT) [17], System Prompts Leaks (SPL) [18], Grok Prompts (GP) [8]. The first three are just collections of leaked or reverse-engineered prompts, similar to CL4R1T4S. The last repository is managed by xAI, which regularly updates the system prompts used for the Grok chat assistant and various product features across X.

Individual observations (i.e., individual system prompts) in each of the selected repositories are stored as files with names that do not always follow a fixed pattern (e.g., `ChatGPT_o3_o4-mini_04-16-2025`, `Claude-2025-05-06`, `claude`). These observations are grouped into folders, most often by model provider (e.g., `ANTHROPIC`, `OPENAI`), but groupings by model family are also possible (e.g., `Claude`, `Grok`). Therefore, the data required preprocessing, ensuring that each observation was associated with a predefined set of metadata: provider, model, and its version. This enabled us to identify corresponding prompts across different datasets. We collected this data through manual processing, most often extracting them from the file path (e.g., `awesome-ai-system-prompts/ChatGPT/4o` was tagged with *OpenAI* (provider), *ChatGPT* (model) and *4o* (version)), and sometimes also from the prompt itself (e.g., `CL4R1T4S/ANTHROPIC/Claude-4.1.txt` does not provide the full version name, which can be retreived from the prompt text: *This iteration of Claude is Claude Opus 4.1 from the Claude 4 model family.*).

Comparing candidate prompts with reference prompts required defining suitable similarity metrics. We first focused on syntactic similarity to detect exact or near-exact matches between candidate system prompts and prompts from other repositories. For each pair of candidate and reference prompts, we computed several statistics. The main metric was an overall similarity score based on a sequence comparison algorithm that identifies the longest contiguous matching subsequences between two texts and applies this procedure recursively to the remaining unmatched parts. The final similarity value is expressed as a normalised ratio between the total length of all matching subsequences and the combined length of both texts[1].

In addition to the overall similarity score, we measured the ratio of identical lines to the total number of lines, word-level similarity using the same sequence-based comparison principle, and the difference in prompt size, defined as the number of characters. The results of the overall similarity analysis are shown in Table 1. The similarity values rarely exceeded 0.5, indicating substantial syntactic differences between the prompts. This outcome is expected given the data collection methodology. Reverse prompt engineering is likely to yield paraphrases of the original system prompt, as a large language model processes and reformulates it. As a result, prompts may convey the same meaning and originate from an identical underlying system prompt, while still differing significantly at the syntactic level.

For this reason, we decided to use the semantic similarity measure—the BERT Score [21]. The results of the precision analysis of this measure are presented in Table 2. As shown, all candidate prompts that had corresponding reference prompts were paired with at least one reference prompt with a satisfactory level of semantic similarity (above 0.5). The analysis described leads us to conclude that the data collected in the candidate repository are reliable, which is why we used this CL4R1T4S dataset in our work.

4 Dataset

The CL4R1T4S dataset [3] is a community-curated collection of system prompts, internal guidelines, and auxiliary instruction texts associated with a wide range of contemporary AI models and agents. Its stated goal is to improve transparency by exposing system-level instructions that are normally inaccessible to end users. The dataset is assembled from leaked and reverse-engineered

[1] The described sequence comparison algorithm was implemented using the `difflib` Python package, which provides a reference implementation of this method [15].

Table 1: Similarity Scores Matrix. The rows are these candidate prompts (identified by provider, model, and its version) that have a corresponding prompt in the reference datasets. Each column corresponds to a reference dataset. If there was more than one observation for a given triple (provider, model and its version), the mean of every pair similarity was calculated.

Provider	Model	Version	AASP	SPL
Anthropic	Claude	Opus 4.1		0.426
Anthropic	Claude	Opus 4.5		0.97
Anthropic	Claude	Sonnet 3.7	0.059	0.052
Anthropic	Claude	Sonnet 4		0.214
Anthropic	Claude	Sonnet 4.5		0.213
Google	Gemini	2.5 Pro		0.015
Google	Gemini	Diffusion	0.764	1
Meta	Llama	4	0.196	
OpenAI	Atlas			0.006
OpenAI	ChatGPT	4.1		0.757
OpenAI	ChatGPT	4.5	0.738	0.518
OpenAI	ChatGPT	4	0.513	0.289
OpenAI	ChatGPT	5	0.329	0.05
OpenAI	ChatGPT	o4 mini	1	0.302
xAI	Grok	3	0.533	

(a) AASP and SPL repositories.

Provider	Model	Version	GP	SPaMoAT
Anthropic	Claude	Sonnet 4.5		0.345
xAI	Grok	3	0.703	
xAI	Grok	4.1	0.156	

(b) GP and SPaMoAT repositories.

Table 2: BERT Scores Precision Matrix. The rows are these candidate prompts (identified by provider, model, and its version) that have a corresponding prompt in the reference datasets. Each column corresponds to a reference dataset. If there was more than one observation for a given triple (provider, model and its version), the mean of every pair of BERT score precision was calculated.

Provider	Model	Version	AASP	SPL
Anthropic	Claude	Opus 4.1		0.791
Anthropic	Claude	Opus 4.5		0.891
Anthropic	Claude	Sonnet 3.7	0.736	0.497
Anthropic	Claude	Sonnet 4		0.52
Anthropic	Claude	Sonnet 4.5		0.626
Google	Gemini	2.5 Pro		0.559
Google	Gemini	Diffusion	0.909	1
Meta	Llama	4	0.711	
OpenAI	Atlas			0.575
OpenAI	ChatGPT	4.1		0.738
OpenAI	ChatGPT	4.5	0.8	0.651
OpenAI	ChatGPT	4	0.733	0.675
OpenAI	ChatGPT	5	0.749	0.571
OpenAI	ChatGPT	o4 mini	1	0.859
xAI	Grok	3	0.666	

(a) AASP and SPL repositories.

Provider	Model	Version	GP	SPaMoAT
Anthropic	Claude	Sonnet 4.5		0.579
xAI	Grok	3	0.691	
xAI	Grok	4.1	0.89	

(b) GP and SPaMoAT repositories.

prompts, which reflect the informal and heterogeneous nature of its sources. At the time of writing, CL4R1T4S remains under active development, with new prompts and revisions added on an ongoing basis.

The dataset covers system prompts associated with 25 companies and projects, including Anthropic, Google, OpenAI, Meta, and Cursor. Prompts are organised as individual files, primarily in text and Markdown formats, grouped into directories that typically reflect model providers or model families. In total, the dataset consists of 59 files with a combined size of approximately 0.99 MB.

4.1 Prompt filtering

As a first preprocessing step, we filtered out prompts and prompt fragments that primarily concern software engineering, code generation, formatting, or tool invocation. Entire prompts were removed for agents whose functionality is explicitly limited to programming-related tasks, namely Cline, Cursor, Devin, Replit, SameDev, Vercel V0, and Windsurf.

For the remaining prompts, we applied fragment-level filtering to remove sections that are not relevant to the analysis of linguistic structure and institutional content. This included text related to tool definitions and APIs, such as schema specifications, function signatures, parameter descriptions, and explicit rules governing tool invocation. We also excluded instructions on code generation and code formatting, including references to programming languages, frameworks, and output styling conventions. In addition, formatting and markup guidelines (e.g., rules for Markdown, HTML, LaTeX, or citation formatting), instructions for generating files or structured artefacts, and extended examples of correct or incorrect agent responses were removed.

This filtering step was intended to reduce noise introduced by highly technical or procedural content and to ensure that the retained text primarily reflects normative, descriptive, and identity-defining aspects of system prompts. The resulting dataset is therefore more suitable for syntactic decomposition and Institutional Grammar-based analysis of system-level instructions.

4.2 Exploratory Data Analysis

After preprocessing, we have performed exploratory data analysis on the prompts we want to analyse further. Figure 1 shows the number of prompts per organisation. As we want to mainly focus on the biggest AI companies, such as OpenAI, Anthropic, xAI, Google and Meta, it is convenient for us that we have the most prompts from these companies, as it will allow for more extensive analysis. For most companies, we have one prompt (corresponding to one model), so possible directions of analysis are rather limited.

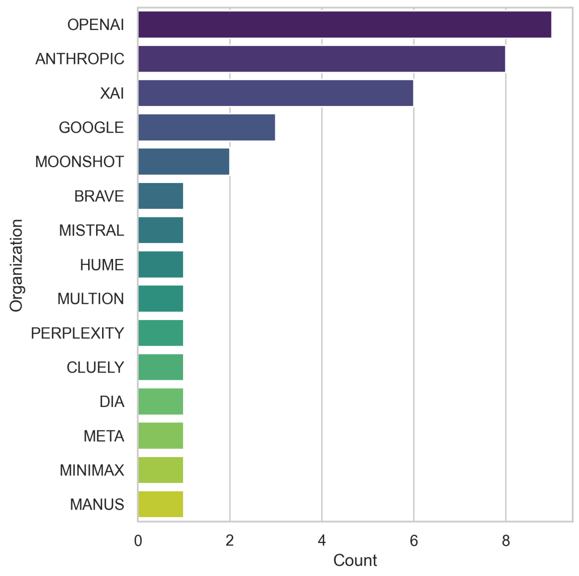


Figure 1: Number of system prompts per organisation

Distribution of the companies' prompts by length is shown in Figure 2. It is evident that the longest prompts are produced by Anthropic, whereas OpenAI's prompts exhibit the widest range of lengths. xAI, Google, and Meta have prompts of similar length, each close to a thousand words.

The length of the Anthropic prompts explains why the word cloud for the dataset shown in Figure 3 shows *Claude* as the most frequent word. Considering it a placeholder for the agent's name, we can see that the prominent words define who the agent is and who the user is (*user, person*). Other most prominent words refer to the instructions, either to responses themselves (*provide, answer, response*) or to the way the responses should be formed (*using, use, information*).

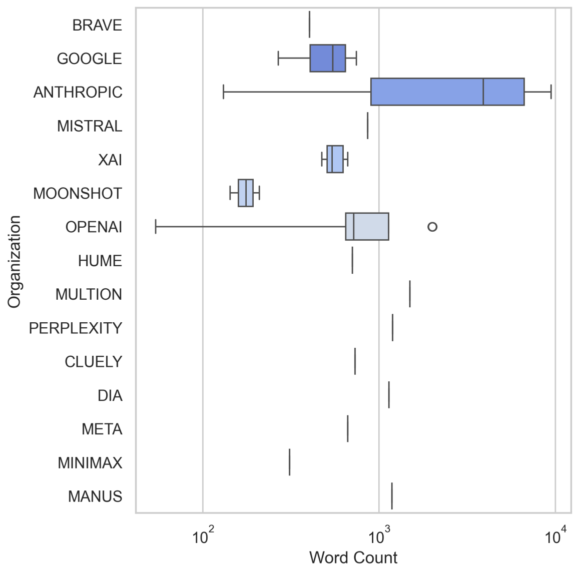


Figure 2: Distribution of system prompt lengths (determined by word count) by organisation

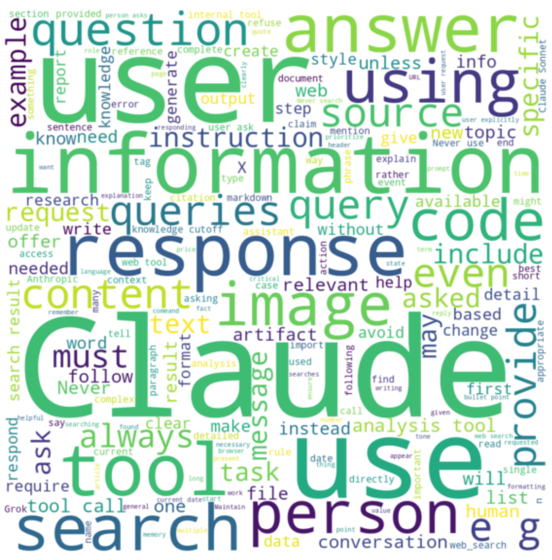


Figure 3: Word cloud of the whole dataset, with the stop words removed

We have also analysed the most common bigrams, concluding that the most frequent (over 80 occurrences) is *analysis tool*, but all of its uses stem from the Claude prompt (in two different versions). Other interesting entries of the most common bigrams are *user asks, person asks, unless user, user explicitly* (all appear between 20 and 40 times), which might correspond with defining constraints for the model and making sure it stays on the right track, responding only to the asked questions and avoiding hallucinations.

5 Statement classification and IG tagging

5.1 Statement classes

To enable a systematic analysis of leaked system prompts, we first introduce a categorisation of statements. This classification is a necessary prerequisite for applying Institutional Grammar (IG), since different types of statements require distinct analytical treatments. We distinguish three mutually exclusive statement classes: (i) non-statements, (ii) regulative statements, and (iii) constitutive statements.

Non-statements are textual segments that do not express institutional meaning in the sense of IG. This category includes descriptive metadata, headings, formatting artefacts, and explanatory passages that neither define institutional entities nor prescribe, permit, or prohibit actions. Such segments are excluded from IG tagging.

Regulative statements are statements that constrain, permit, or prescribe actions of an actor. In the context of system prompts, regulative statements typically appear as instructions, constraints, or permissions directed at the language model, even when explicit legal or deontic markers are absent.

Constitutive statements, by contrast, establish or modify institutional facts by defining entities, assigning roles, or specifying properties within the system. These statements do not directly regulate behaviour, but instead construct the conceptual and institutional framework within which regulative statements operate. In leaked system prompts, constitutive statements often describe the language model's identity, capabilities, or scope.

5.2 IG annotation rules

Institutional Grammar provides a well-established and expressive framework for analysing institutional statements, particularly in domains with formalised, explicitly structured language, such as statutory law or policy documents. However, leaked system prompts for large language models differ substantially from prompts in these domains. Their language is often informal, underspecified, and strongly context-dependent, which creates challenges for a direct application of the full IG schema. In addition, such texts are not always grammatically correct, and relevant information may be conveyed implicitly or through informal devices, such as capitalisation.

Empirically, we observe many statements that clearly serve a regulative function but omit one or more canonical IG components, for example, an explicit actor or a deontic operator. In other cases, correct interpretation is only possible by incorporating contextual information from surrounding text, such as section headings or enumeration labels. This is particularly common in instructional prompts, where constraints and permissions are distributed across multiple textual units rather than expressed in a single, self-contained sentence. We also observe that a penalty is almost never given for breaking a prohibition. Strict adherence to the full IG specification would therefore lead either to systematic under-annotation or to speculative reconstruction of missing elements.

To address these issues while preserving analytical rigour, we adopt a simplified version of IG that is tailored to the empirical properties of the text. For constitutive statements, we extract three components: Constituted Entity (E), Constitutive Function (F), and Constituting Properties (P). This reduced set captures the minimal structure required to represent institutional definitions. For example, in the statement "The assistant (E) is (F) Claude (P)", the entity is defined through an attributive function and a property.

For regulative statements, we extract Attribute (A), representing the addressee of the statement; Aim (I), denoting the regulated action; Object (B), indicating the entity toward which the action is directed; Deontic (D), expressing the normative modality when explicitly present; and Conditions (C), specifying the contextual constraints under which the regulation applies. For instance, in the statement "If the person asks (C), Claude (A) can (D) tell (I) them about the following products which allow them to access Claude (B)", the regulative structure becomes explicit through these components.

If the Attribute or Constituted Entity is not explicitly stated, we interpret it as "You", since most system prompts directly address the lan-

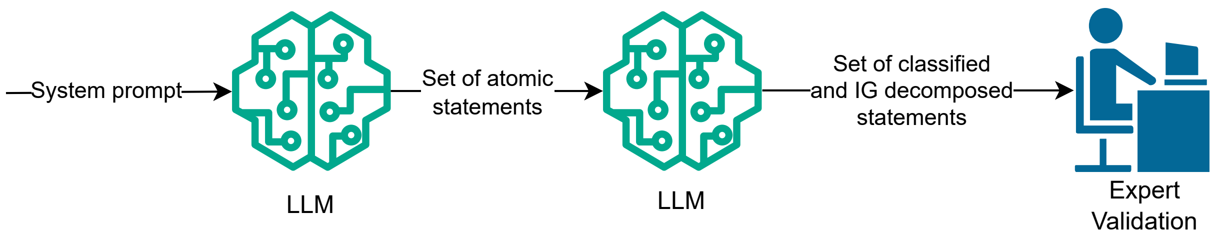


Figure 4: Diagram of processing pipeline.

guage model.

5.3 Statement atomicity

System prompt complexity varies substantially across providers. Some prompts use a single sentence per statement, while others encode multiple complex statements within a single sentence. To allow for fair comparison, we first split the text into atomic statements.

Defining atomicity is not trivial. We apply a practical rule of thumb: a regulative atomic statement contains a single Attribute, Aim, and Object, while a constitutive atomic statement contains a single Constituted Entity, Constitutive Function, and Constituting Properties. Ambiguity arises when determining whether an Object or a set of Properties should be treated as a single entity or further divided. For example, an Object described by multiple adjectives may arguably form more than one statement.

In such uncertain cases, we prefer a more granular split rather than a coarser one. When ambiguity cannot be resolved, we split statements into smaller atomic units rather than risk undersegmentation.

5.4 Processing pipeline

The primary objective of this study is the qualitative and structural analysis of leaked system prompts using statement classification and IG annotation, rather than the development or evaluation of fully automated preprocessing methods. Consequently, data quality is prioritised over scalability. At the same time, fully manual preprocessing and annotation of the entire dataset were infeasible due to limited team size and the domain expert's restricted availability.

To address this constraint, we adopt an LLM-accelerated manual annotation strategy. In this approach, large language models generate structured annotation proposals, while final decisions remain under expert control. Expert validation is essential for reliably capturing subtle institutional meanings, especially in cases involving implicit actors, context-dependent norms, or fragmented statement structures. Although cross-validation by multiple experts would be methodologically preferable, it was not feasible within the scope of this project.

The processing pipeline consists of three clearly separated stages, as illustrated in Figure 4.

In the first stage, the preprocessed system prompt text is provided to a language model. Tool- and code-related fragments (described in Section 4.1) are filtered beforehand. The model is instructed to extract all statements from the text and split them into atomic statements where necessary. Importantly, the model must also attach relevant contextual information for correct interpretation. This includes, in particular, enumeration headings, which frequently encode shared conditions, scopes, or constraints that apply to all listed statements. When an actor or other required element is not explicitly stated, the model is instructed to make it explicit.

The output of the first stage is a set of statements for each system prompt. Each entry contains a statement anchor, an atomic statement rewritten in grammatically correct form, and the associated contextual elements required for interpretation.

In the second stage, each extracted statement is processed independently. The statement is classified into one of the predefined statement classes and annotated using the simplified IG schema described above. At this stage, the language model proposes the statement class and the corresponding IG components based on both the statement text and its attached context. The result is a set of structured objects that conform to a fixed annotation schema, ensuring consistency across the dataset.

In the final stage, all automatically generated classifications and IG annotations are subjected to

Preprocessed System Prompt:
```
When writing a single reply, follow these rules:
- Incorporate all specific tone or content information provided by the user into the
reply.
...
```

Extracted Statement with Context (Atomic Statement 1):

Key	Value
statement	Incorporate all specific tone
with-context	When writing a single reply, you incorporate all specific tone information provided by the user into the reply.

Classified and IG-Tagged Statement (Atomic Statement 1):

Field	Value	Description
full_statement	When writing a single reply, you incorporate all specific tone provided by the user into the reply.	Original full text of the statement
class	regulative	Statement class: regulative, constitutive, or non-statement
A	you	Addressee of the statement (agent performing the action)
I	incorporate	Aim: the regulated action of the addressee
B	all specific tone provided by the user into the reply	Object: the entity affected by the action
D		Deontic modality (implicit obligation)
C	When writing a single reply	Condition or trigger for the regulated action

Extracted Statement with Context (Atomic Statement 2):

Key	Value
statement	or content information provided by the user into the reply
with-context	When writing a single reply, you incorporate all specific content information provided by the user into the reply.

Classified and IG-Tagged Statement (Atomic Statement 2):

Field	Value	Description
full_statement	When writing a single reply, you incorporate all specific content information provided by the user into the reply.	Original full text of the statement
class	regulative	Statement class: regulative, constitutive, or non-statement
A	you	Addressee of the statement (agent performing the action)
I	incorporate	Aim: the regulated action of the addressee
B	all specific content information provided by the user into the reply	Object: the entity affected by the action
D		Deontic modality (implicit obligation)
C	When writing a single reply	Condition or trigger for the regulated action

Figure 5: Example of processing a system instruction into atomic regulative statements and structured IG fields. Tables visualise JSON objects.

expert validation. A human annotator with expertise in Institutional Grammar reviews each statement using a dedicated web application developed for this study. The application supports systematic inspection, correction, and confirmation of extracted components, ensuring that all final annotations reflect expert judgment.

An example of a statement at different stages of the processing pipeline is shown in Figure 5. The exact instructions used for the LLM-based stages are documented in Appendix B, while the design and functionality of the validation application are described in Appendix C.

5.5 Incidental observation on system prompt behaviour

During the study, we observed an unexpected behaviour when using the first system prompt that instructed preprocessing, as described in Appendix B. When the prompt was provided to certain large commercial language models (ChatGPT 5 Mini, Gemini 3.5 Flash) without explicit input text, they began to parse their own system prompt (or make something up, as its authenticity cannot be fully verified) instead of waiting for external input.

This behaviour was discovered incidentally during the preparation of test cases for manual annotation. While the original goal was to extract statements from leaked system prompts, this observation suggests that the initial prompt can serve as a mechanism for extracting system prompts.

We stress that this finding is preliminary and anecdotal. The reliability and generalizability of this behaviour across different models have not been systematically evaluated. Therefore, it should not be considered a validated method for extracting system prompts.

6 Analysis

This section presents a multi-faceted quantitative analysis of system prompt evolution, structure, and semantic relationships across major LLM providers. We examine longitudinal trends in prompt characteristics, provider-specific linguistic patterns, deontic governance strategies, and semantic clustering.

6.1 Prompts' Evolution Analysis

To analyse prompt evolution across the primary models, we plotted changes in Size (S), measured as character count, the Non-Code-to-Whole (NCtW) ratio, defined as the proportion of non-code-related content relative to the entire prompt, and Constitutive-Statements-to-All (CStA) ratio, calculated as the proportion of constitutive statements to all statements, across subsequent system prompt versions. Figure 6 illustrates the character counts for these prompts over time. We observe a general upward trend, indicating that system prompts are becoming increasingly longer. This aligns with expectations: as models evolve, instructions become more precise, and additional tools and their descriptions are integrated into the system. This growth is most prominent in Anthropic's Claude models, for which we had the most extensive data; in this case, the increase in prompt size is particularly clear.

Based on the analysis of the prompt NCtW evolution over time (Figure 7), we observe a more complex dynamics compared to the straightforward increase in prompt size. However, a general declining tendency in this ratio appears to

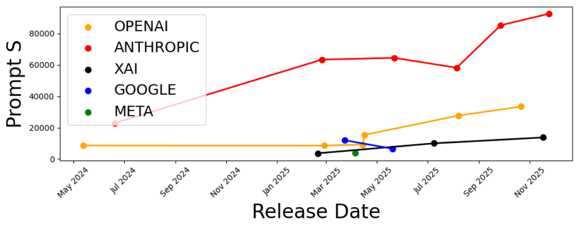


Figure 6: Prompt size (in characters) over time

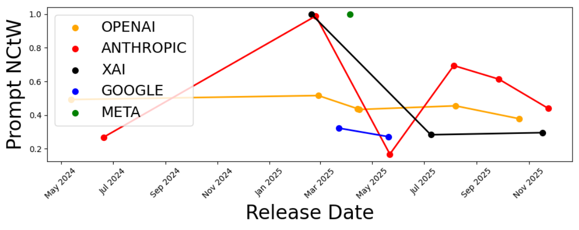


Figure 7: Prompt NCtW over time

emerge over time. This suggests that while system prompts are indeed growing longer overall, the proportion of that growth dedicated to non-code, instructional content may be decreasing relative to code-related components. It is worth noting that we have considered "code" fragments both for output code formatting and for references and examples of available tools and APIs. This decline might mean that, over time, agents have more and more tools available to them.

Figure 8 shows the evolution of the CStA ratio. The underlying hypothesis was that the development of the non-coding parts of the system prompts would occur primarily through an expansion of regulative statements, as these are the main mechanisms used to instruct the model on how to behave. Consequently, we expected CStA to decrease over time, reflecting a growing emphasis on prescriptive, rule-like guidance relative to descriptive or capability-defining content.

The empirical results do not support this hypothesis straightforwardly. Rather than a clear monotonic decline, the CStA ratio exhibits a nonlinear trajectory over successive prompt versions. There is a period during which the overall trend appears to be decreasing, which is consistent with the hypothesised shift towards more regulative content. However, in the most recent versions, this tendency reverses, and the CStA ratio increases again. This pattern suggests that prompt designers alternate between phases of adding more detailed behavioural constraints and phases of rebalancing or restructuring prompts towards more con-

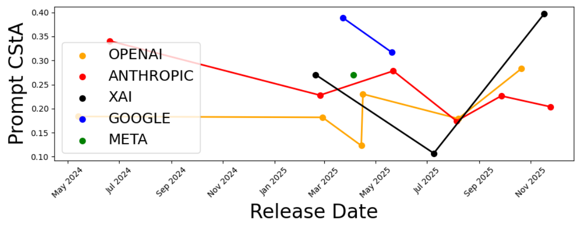


Figure 8: Prompt CStA over time

stitutive formulations, possibly to improve clarity, maintainability, or model usability.

6.2 Prompt Analysis per Provider

Given the observed growth in system prompt length, we aimed to analyse the textual complexity of these prompts. In particular, we investigated whether different providers adopt distinct strategies in terms of using more complex or more easily readable prompts, and whether prompt complexity is correlated with prompt length.

To quantify textual complexity, we used the Flesch Reading Ease (FRE) score as a proxy, defined as:

$$\text{FRE} = 206.835 - 1.015 \times \left(\frac{\text{words}}{\text{sentences}}\right) - 84.6 \times \left(\frac{\text{syllables}}{\text{words}}\right) \text{ [5].}$$

The FRE score ranges from 0 to 100, with lower values indicating more difficult-to-read text and higher values indicating simpler, more readable text. We selected FRE because it is independent of a specific LLM, tokeniser, or training dataset, making it suitable for cross-provider comparisons.

In general, short, direct instructions tend to yield higher FRE scores, reflecting lower complexity, whereas longer prompts containing conditional statements and abstract instructions tend to yield lower FRE scores, indicating higher complexity.

As shown in Figure 9, Anthropic and Google use prompts with FRE scores around 40–45, corresponding to a college-level reading difficulty. The difference between these providers is that Anthropic tends to use significantly longer prompts (up to approximately 6000 characters), whereas Google uses shorter prompts (up to approximately 1000 characters). OpenAI uses moderately complex, medium-length prompts. Meta and XAI use prompts with FRE scores around 50, corresponding to high school-level reading difficulty.

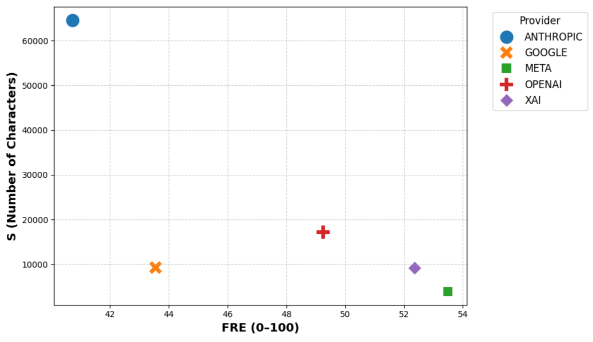


Figure 9: Prompt Size vs Flesch Reading Ease per Provider

6.2.1 Deontics analysis

As a next step, the regulative statements were analysed according to their deontic type. Commands were defined as regulative statements containing deontic expressions such as “must” or “should”, permissions as statements using “may” or “can”, and prohibitions as statements using “must not” or “may not”. In addition, we distinguished a fourth category, strategies, which capture suggestions about the agent’s behaviour that typically do not contain any explicit deontic marker.

Table 3 summarises this classification scheme. For each system prompt, we computed the following indicators: the proportion of commands to all regulative statements (CtRS), the proportion of permissions to all regulative statements (PRtRS), the proportion of prohibitions to all regulative statements (PHtRS), and the proportion of strategies to all regulative statements (StRS).

Table 3: Regulative statements classification by deontics

Category	**Deontic Keywords**
Command	must, should, has to, will
Permission	may, can
Prohibition	cannot, not (e.g., must not)
Strategy	no explicit deontic

The distribution of these indicators across selected providers is shown in Figure 10. Anthropic and OpenAI show a strong preference for Strategy statements, indicating that those providers rely on a ”Descriptive” rather than a ”Direct” governance style, which might increase the likelihood of hallucinations.

For further analysis, the StRS was removed

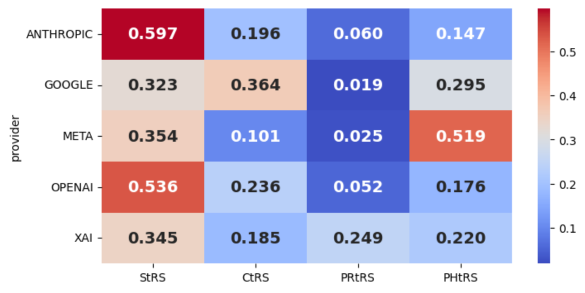


Figure 10: Heatmap of statements classified according to the deontic classification scheme.

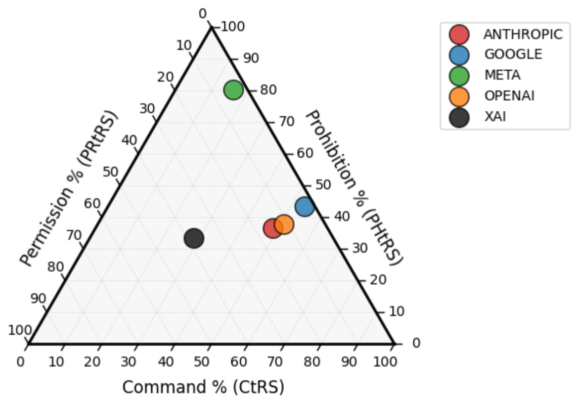


Figure 11: Ternary plot - Command, Permission, Prohibition per Provider

since, from the linguistic point of view, Strategies were less valuable. The rest of the measures were normalised. Results were plotted as a ternary plot and are visible in Figure 11. On the plot, 3 clusters are visible:

- **xAI**: The most diverse distribution, showing approximately 30% prohibition and 30% command, permissions around 40%.
- **Meta**: utilizing permission ($\approx$ 5%), prohibition ($\approx$ 80%), and command sentences ($\approx$ 15%).
- **OpenAI, Google, Anthropic**: A cluster characterised by negligible use of permission statements and slightly varied usage of prohibition ($\approx$ 40%), but extensive usage of command (50%).

We tested the hypothesis that Grok's system prompts, and thus xAI's models, are the most permissive, motivated by the provider's emphasis on openness and reduced constraints. We therefore expected xAI to show relatively low levels of

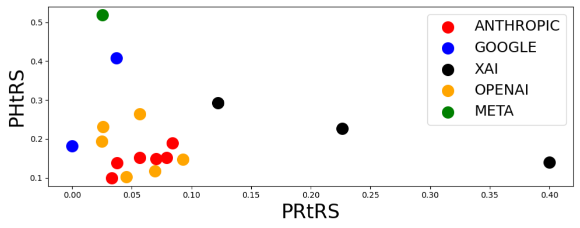


Figure 12: The proportion of prohibitions (PHtRS) against the proportion of permissions (PRtRS) to all regulative statements.

prohibitions (PHtRS) and high levels of permissions (PRtRS). To evaluate this, we plotted PHtRS against PRtRS for all providers (Figure 12). xAI occupies a central position in the plot, reflecting a balanced distribution of prohibitions and commands and a higher share of permissions, indicating a diverse governance style rather than a purely permissive one.

Meta, by contrast, is dominated by prohibitions, with limited use of commands and permissions, suggesting a strongly restrictive governance strategy. OpenAI, Google, and Anthropic form a cluster characterised by extensive use of commands, moderate prohibitions, and negligible permissions, reflecting a directive but not exclusively prohibitive governance approach.

In summary, the deontic analysis reveals clear differences in how providers structure their regulative statements. xAI demonstrates the most diverse distribution across prohibitions, commands, and permissions, indicating a complex governance approach that combines constraints with allowances. Meta exhibits a predominantly restrictive strategy centred on prohibitions, whereas OpenAI, Google, and Anthropic adopt a command-oriented style that is directive yet not exclusively prohibitive.

6.3 Prompt-Space Visualizations

To explore semantic relationships between system prompts and construct a comprehensive "prompt-space" mapping, we implemented text embeddings using the SentenceTransformer model with the `all-MiniLM-L6-v2` architecture [16]. Sentence-BERT addresses the computational inefficiency of pairwise sentence comparisons by introducing network architectures that produce fixed-dimensional sentence embeddings. It employs pooling strategies over token representations, with mean pooling proving most effective,

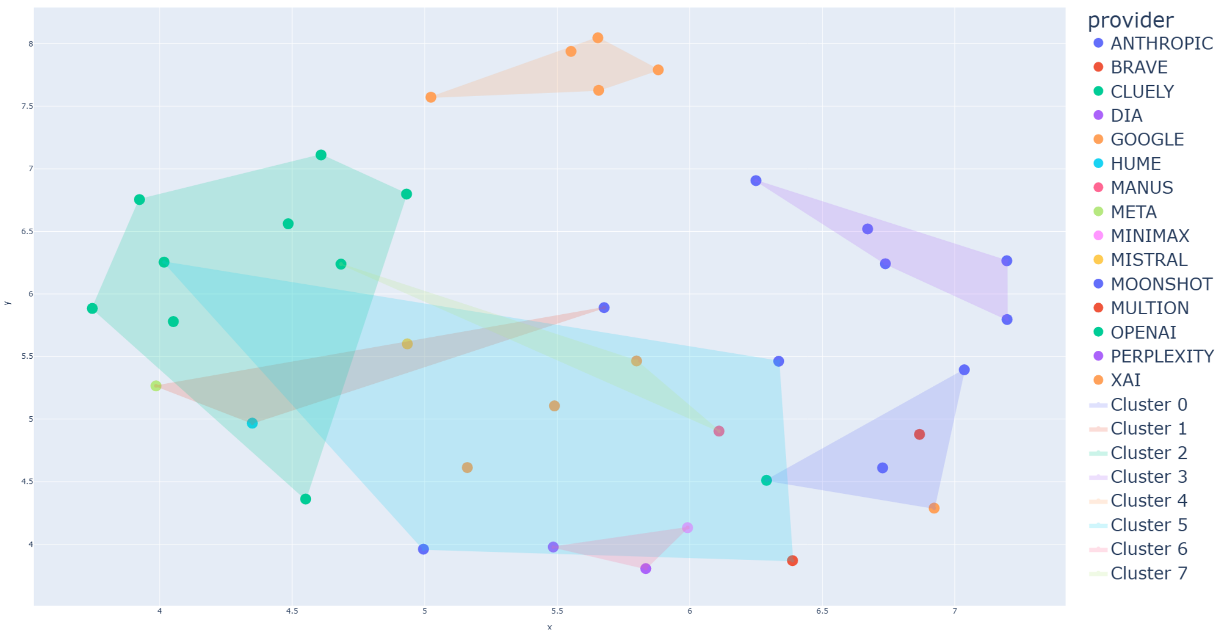


Figure 13: Sentence-BERT prompts embeddings in 2D space. Clusters were determined using the Spectral algorithm and the elbow method.

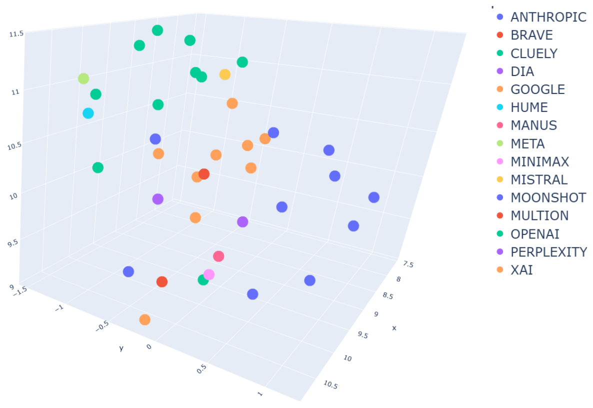


Figure 14: Sentence-BERT prompts embeddings in 3D space.

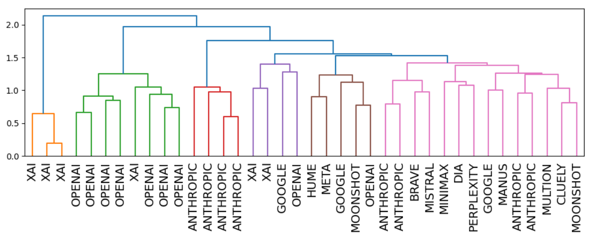


Figure 15: Hierarchical clustering dendrogram of Sentence-BERT prompts embeddings.

enabling efficient semantic similarity computation via cosine similarity. This approach converts entire system prompts into high-dimensional vector representations that capture their semantic content, enabling quantitative comparison across different providers and model generations.

Note that while previous sections of analysis in this chapter focused on a selected subset of providers – prioritising those most substantially represented in the dataset with possibly multiple prompts to enable evolution pattern analysis – this embedding analysis incorporates prompts from all available providers. The embeddings were first clustered using the Spectral algorithm, with the optimal number of clusters determined via the elbow method, revealing 8 distinct groups. To facilitate visualisation, we applied UMAP dimensionality reduction [10] to project these embeddings into 2D (Figure 13) and 3D (Figure 14) spaces. The results demonstrate clear separation between some providers: most OpenAI prompts form a cohesive cluster, while xAI prompts occupy a distinct region easily distinguishable from others. This provider-specific clustering suggests that these companies maintain consistent semantic ”signatures” in their system prompt design.

To further explore hierarchical relationships and construct a ”family tree” of system prompts, we applied agglomerative hierarchical clustering with Ward linkage, visualised as a dendrogram (Figure 15). This analysis confirms the 2D/3D clustering pattern – most OpenAI prompts form almost their own major branch.

6.4 Analysis Summary

The analysis reveals several key patterns in system prompt design. First, prompt evolution shows a clear trend toward increasing length over time, particularly evident in Anthropic's Claude lineage, driven largely by the expansion of code- and tool-related content. However, the relative proportions of non-code instructions (NCtW) exhibit a declining tendency, while the proportion between constitutive and all statements (CStA) follows a non-monotonic trajectory. These findings suggest an ongoing iterative redesign process rather than stable directional evolution, with more extensive longitudinal data needed for robust generalisation.

Second, textual complexity, measured via the Flesch Reading Ease score, varies significantly across providers. Anthropic and Google use prompts of college-level difficulty, whereas Meta and xAI use more readable, high school-level instructions. Notably, prompt length does not always correlate directly with textual complexity. For example, Anthropic's long prompts maintain high complexity, while Google achieves similar complexity with significantly shorter prompts. This indicates that different providers adopt distinct strategies to balance clarity, instruction specificity, and cognitive load for the model.

Third, the deontic analysis highlights diverse governance strategies across providers. Meta demonstrates a prohibitive approach, with a high proportion of prohibition statements; xAI balances commands, permissions, and prohibitions more evenly; and OpenAI, Google, and Anthropic cluster together, with strong command and prohibition usage but minimal permissions. These patterns suggest that providers vary in how they enforce behavioural constraints on models, with some emphasising flexibility and others prioritising directive control.

Finally, the prompt-space visualisations using Sentence-BERT embeddings demonstrate that full-prompt semantic representations form provider-specific clusters, with OpenAI and xAI exhibiting particularly cohesive and distinguishable groupings. While these results validate the existence of company-specific semantic signatures, the separation is less pronounced for some providers, likely due to limited data availability – small numbers of prompts per organisation disturb robust identification of repeated patterns across all cases.

7 Conclusions and Future Work

This study provides a systematic analysis of leaked LLM system prompts using a simplified Institutional Grammar framework, revealing trends in their structural evolution and provider-specific design patterns. Prompts have grown substantially longer over time, with a relative decline in non-code instructional content in favor of introducing various tools available for the agents to use, while textual complexity varies across providers. Deontic analysis highlights diverse governance styles: xAI's balanced approach, Meta's prohibition-heavy restrictions, and the command-dominant strategies of OpenAI, Google, and Anthropic.

The LLM-assisted pipeline, as well as the annotated dataset produced by it, can serve as a foundation for developing improved annotation processes, enabling more scalable decomposition of future prompt leaks. As additional data accumulates in repositories like CL4R1T4S, the analysis can be repeated to confirm or refine observed trends in prompt evolution and provider differences. Following suggestions from peer reviews, empirical studies could test how structural features—such as deontic ratios of commands, prohibitions, and permissions—affect model behaviors.

References

[1] *Awesome AI System Prompts*. Accessed on 16.12.2025. 2025. URL: https://github.com/dontriskit/awesome-ai-system-prompts.

[2] Gaurav Beri and Vaishnavi Srivastava. "Advanced Techniques in Prompt Engineering for Large Language Models: A Comprehensive Study". In: *2024 IEEE 4th International Conference on ICT in Business Industry Government (ICTBIG)*. 2024, pp. 1–4. DOI: 10.1109/ICTBIG64922.2024.10911672.

[3] *CL4R1T4S: Leaked system prompts for major AI models*. Accessed on 16.12.2025. 2025. URL: https://github.com/elder-plinius/CL4R1T4S.

[4] Susan E. S. Crawford and Elinor Ostrom. *A Grammar of Institutions*. Washington, D.C.: American Political Science Association, 1995. ISBN: 978-0917495564.

[5] Rudolph Flesch. “A new readability yardstick”. In: *Journal of Applied Psychology* 32.3 (1948). Place: US Publisher: American Psychological Association, pp. 221–233. ISSN: 1939-1854. DOI: 10 . 1037 / h0057532.

[6] Christopher K. Frantz and Saba Siddiki. “Institutional Grammar 2.0: A specification for encoding and analyzing institutional design”. In: *Public Administration* 99.2 (2021), pp. 222–247. DOI: 10.1111/padm.12719.

[7] Christopher K. Frantz and Saba Siddiki. *Institutional Grammar: Foundations and Applications for Institutional Analysis*. Cham, Switzerland: Springer Nature, 2022. ISBN: 978-3-030-86371-5. DOI: 10.1007/978-3-030-86372-2.

[8] *Grok Prompts*. Accessed on 16.12.2025. 2025. URL: https://github.com/xai-org/grok-prompts.

[9] Qianou Ma et al. “What Should We Engineer in Prompts? Training Humans in Requirement-Driven LLM Use”. In: *ACM Trans. Comput.-Hum. Interact.* 32.4 (Aug. 2025). ISSN: 1073-0516. DOI: 10 . 1145 / 3731756.

[10] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[11] Murimo Bethel Mutanga et al. “Exploring the Impact of LLM Prompting on Students’ Learning”. In: *Trends in Higher Education* 4.3 (2025). ISSN: 2813-4346. DOI: 10.3390/higheredu4030031.

[12] Anna Neumann et al. “Position is Power: System Prompts as a Mechanism of Bias in Large Language Models (LLMs)”. In: *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’25. Association for Computing Machinery, 2025, pp. 573–598. ISBN: 9798400714825. DOI: 10 . 1145 / 3715275.3732038.

[13] Marco Polignano, Marco De Gemmis, and Giovanni Semeraro. “Unraveling the Enigma of SPLIT in Large-Language Models: The Unforeseen Impact of System Prompts on LLMs with Dissociative Identity Disorder”. In: *Proceedings of the Tenth Italian Conference on Computational Linguistics (CLiC-it 2024)*. Ed. by Felice Dell’Orletta et al. Pisa, Italy: CEUR Workshop Proceedings, Dec. 2024, pp. 774–780. ISBN: 979-12-210-7060-6.

[14] *Proof of Concept: Dangers of System Prompt Leakage*. Tech. rep. RESK, 2025. URL: https://resk.fr/pdf/resk-enterprise-ai-security-guide.pdf.

[15] Python Software Foundation. *difflib — Helpers for computing deltas*. Python Standard Library documentation. 2024.

[16] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL].

[17] *System Prompts and Models of AI Tools*. Accessed on 16.12.2025. 2025. URL: https://github.com/x1xhlol/system-prompts-and-models-of-ai-tools.

[18] *System Prompts Leaks*. Accessed on 16.12.2025. 2025. URL: https://github.com/asgeirtj/system_prompts_leaks.

[19] J.D. Zamfirescu-Pereira et al. “Why Johnny Can’t Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI ’23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9781450394215. DOI: 10.1145/3544548.3581388.

[20] Collin Zhang, John Xavier Morris, and Vitaly Shmatikov. “Extracting Prompts by Inverting LLM Outputs”. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 14753–14777. DOI: 10.18653/v1/2024.emnlp-main.819.

[21] Tianyi Zhang et al. “BERTScore: Evaluating Text Generation with BERT”. In: Eighth International Conference on Learning Representations. Apr. 2020. URL: https://iclr.cc/virtual_2020/poster_SkeHuCVFDr.html (visited on 01/25/2026).

[22] Mingqian Zheng et al. “When “A Helpful Assistant” Is Not Really Helpful: Personas in System Prompts Do Not Improve Performances of Large Language Models”. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 15126–15154. DOI: 10.18653/v1/2024.findings-emnlp.888.

A Work division

Table 4: Author Contributions

Task	MM	JD	WG	WM
Problem formulation and study design	✓	✓	✓	✓
Literature review	✓		✓	
Dataset selection and preprocessing	✓	✓	✓	
Evaluation methodology	✓	✓	✓	✓
Exploratory Data Analysis (EDA)	✓	✓	✓	✓
Pipeline methodology	✓			
Pipeline implementation	✓			
Results analysis			✓	✓
Visualization and plots			✓	✓
Report writing	✓	✓	✓	✓
Reviews and review rebuttals		✓		✓
Editing and final review	✓	✓	✓	✓

MM – Marek Mytkowski,
WG – Weronika Gozdera,
JD – Julia Dudzińska,
WM – Wojciech Michaluk

B LLM System Prompts Used in the Processing Pipeline

This appendix documents the system prompts used in the large language model (LLM)–based stages of the processing pipeline. The purpose of including these prompts is to ensure methodological transparency and reproducibility, as the behavior of LLMs is highly sensitive to prompt formulation. All prompts reported here were used consistently across the entire dataset and were not adapted on a per-document basis.

The prompts are organized according to the corresponding pipeline stages. For each prompt, we provide its functional role within the pipeline and the exact textual formulation supplied to the LLM.

B.1 Statement Extraction Prompt

This prompt is used in the first stage of the pipeline, where the objective is to extract all statements from a preprocessed system prompt. The model is instructed to identify not only standalone statements but also to include any relevant contextual elements, such as enumeration headings or section titles, whenever they are required for correct semantic or institutional interpretation.

Purpose. The goal of this prompt is to obtain an exhaustive and context-aware segmentation of the input text into analyzable statement units. This step is critical, as missing contextual information would lead to systematic errors in subsequent classification and IG tagging.

System prompt.

You are a high-precision linguistic preprocessor. Your task is to decompose complex LLM system instructions into **atomic, independent statements**.

I. Core Objective: Atomic Decomposition

An *atomic statement* contains exactly one **Attribute** (You, person about which this statement is), **Deontic** (if present, must, can, etc.), **Aim** (verb, action, for example "follow"), and **oBject** (rules). It contains exactly one primary claim, identity, or rule.

If a sentence contains multiple adjectives, qualities, or actions separated by commas or conjunctions, split each into separate statements, even if they belong to the same sentence. You **must split** compound sentences or lists into separate JSON objects.

- **Split on Conjunctions:** If a sentence says "Do X and Y," create two objects: one for "Do X" and one for "Do Y."
- **Split on Punctuation:** Semicolons (;) and distinct clauses must be treated as separate statements.
- **Split Compound Predicates:** If a sentence contains multiple verbs or actions joined by "and", "or", or commas, each verb/action must become a separate atomic statement, even if it shares the same subject.
- **Split Complex Clauses:** If a sentence contains multiple clauses with different subjects, objects, or actions, split into separate atomic statements.

II. Field Definitions

1. **statement (Verbatim):** The exact, original text from the prompt. If you split a sentence, this field contains the specific segment assigned to this object.
2. **with-context (Self-Contained Paraphrase):** A reconstructed version of the statement that incorporates headers, list titles, or conditional triggers. It should be a grammatically complete sentence that can be understood in isolation. Extract the explicit actor from the statement and context; if it is not present, use the implicit actor "You". Do not use the third person if it was not used in the statement. You must not use deontics (e.g. "must", "should", "can") if they were not present in the original statement. You must use them, though, if they are present. Try to use precisely the same wording, though make it grammatically correct and add context.

- **Requirement:** If the statement is under a header like "Safety Rules," the `with-context` should start with "Under Safety Rules..." or incorporate the header's intent. Do not make up headers. Every specific rule, even if related to the same topic (e.g. arithmetic calculation, attention to wording), must become its own atomic statement with its own `with-context`.
- **Preserve Atomicity:** Do not combine multiple ideas in a single statement. If necessary, repeat context to maintain independence.
- **Anchor Headers/Context:** Every atomic statement must reference the section, numbered list, or header it originates from in the `with-context` field.
- **Verification Step:** After decomposition, review each statement to ensure it contains only one primary claim or action and conditions.

III. Instructions

1. **Extract All Statements:** Include institutional (identity/tools), guidelines (principles), and non-institutional (formatting/behavioral) statements.
2. **Decompose Aggressively:** Ensure each statement has only one **Aim** (action), **Attribute**, and **oBject**. Conditions do not count toward these.
3. **Incorporate Metadata:** Use headers, numbered list titles, and global conditions to "anchor" the statement in the `with-context` field. Do not make up metadata; they must be based on document structure (e.g. headers or tags).
4. **Ignore Examples:** Skip illustrative "User/Assistant" chat logs unless they contain a specific rule not mentioned elsewhere. Ignore meta-examples or illustrative text unless they introduce a rule or requirement not stated elsewhere.

IV. Complete Example

Input text:

```
You are Claude, a model trained by Anthropic. You are a
chat model and do not have a hidden chain of thought or
private reasoning tokens. ...
```

Output JSON:

```
[
  {
    "statement": "You are Claude",
    "with-context": "You are Claude."
  },
  {
    "statement": "a model trained by Anthropic.",
    "with-context": "You are a model trained by Anthropic."
  },
  {
    "statement": "You are a chat model",
    "with-context": "You are a chat model."
  },
  {
    "statement": "and do not have a hidden chain of thought
                  or private reasoning tokens.",
    "with-context": "You do not have a hidden chain of thought."
  },
  {
    "statement": "or private reasoning tokens.",
    "with-context": "You do not have a private reasoning tokens."
  }, ...
]
```

You must output only valid JSON. You must not stop until the result is a proper JSON array. You must not stop until the whole system prompt is processed. If you want to stop, check if it is the end of the system prompt.

B.2 Statement Classification and Simplified IG Tagging Prompt

This prompt is used in the second stage of the pipeline. Its purpose is to classify each extracted statement into one of the predefined statement classes and to extract the corresponding components according to the simplified Institutional Grammar schema.

Purpose. The prompt guides the model to (i) take into account both the statement and its extracted context, (ii) assign a statement class, and (iii) identify and output the relevant IG components in a structured format. The output is constrained to a fixed JSON schema to ensure consistency and machine-readability.

System prompt.

You are a strict linguistic analyst specializing in the classification of LLM system instructions. Your task is to extract the semantic components from each statement based on its grammatical and functional role.

Input Format

Your input is a **JSON array of objects** in the following format:

```
[
    {
        "statement": string,
        "with-context": string
    }
]
```

The `"statement"` value is not of your concern and must be returned unchanged in the results as `"anchor"`. The `"with-context"` value is the statement to be analysed and must be returned as `"full_statement"`.

Your output must be a **JSON array of objects** strictly adhering to the schema defined below.

I. The Classification Logic (The Litmus Test)

Determine the `class` of a statement by applying the following tests **in order**.

1. Non-Statement

A statement is a **Non-statement** if it is a dynamic context injection, a timestamp, or a declaration of environmental variables rather than a rule or identity definition.

Examples:

- "The current date is Thursday, May 22, 2025."
- "Current location: Warsaw."

2. Regulative

A statement is **Regulative** if it imposes a rule, constraint, or command on the agent's behavior.

- **Behavioral Constraint Test:** Does the statement prescribe *what the LLM must do or say* (Aim) toward a specific target (Object), particularly in response to a prompt or condition?
- **Conditional Response Block:** If the statement provides facts or data gated by a condition, it is Regulative, as it prescribes the act of using that data.
- **Command/Modal Check:** Does it use an imperative verb (e.g. "Do not", "Refuse") or a modal verb (e.g. "must", "should", "can")?

3. Constitutive

A statement is **Constitutive** if it defines the agent's identity, nature, or inherent, *non-behavioral* facts.

- **Identity/Fact Test:** Does it define *who* the agent is (Entity) via a linking verb (Function) and their nature (Properties)?
- **Unconditional Fact Check:** If a statement is a pure fact with no explicit or implicit command to use it, and no condition, it is Constitutive.

If no class matches, **default to Non-statement**.

II. Field Extraction Rules

Depending on the assigned `class`, extract the following components.

A. Non-Statement

Return all structural fields as `null` or empty strings.

B. Constitutive

Extract the following **Constitutive Syntax** elements:

- **Entity (E):** The subject being defined (e.g. "The assistant", "You", "Grok").
- **Function (F):** The verb connecting the entity to its definition (e.g. "is", "represents", "was created").
- **Properties (P):** The definition, origin, or nature assigned to the entity (e.g. "Claude", "an AI created by Anthropic").

C. Regulative

Extract the following **Regulative Syntax** elements:

- **Attribute (A):** The addressee or actor of the statement (e.g. "You", "The model", "Claude"). If implied by an imperative, use "You".
- **Deontic (D):**
  - Extract a deontic term **only if it explicitly appears** (e.g. "must", "can", "should", "may").
  - If the statement is imperative (e.g. "Refuse to…"), set `Deontic` to an empty string.
- **Aim (I):** The action verb describing what the Attribute does (e.g. "tell", "refuse", "prioritize").
  - **Handling Prohibitions:** If avoidance is implied (e.g. "Do not hallucinate"), incorporate negation into the Aim (e.g. "not hallucinate").
- **Object (B):** The target of the Aim (e.g. "questions about elections", "users").
- **Conditions (C):** Specific scopes, triggers, or exceptions (e.g. "If the person asks", "unless explicitly requested"). If the rule is general, return an empty string.

III. Output JSON Schema

Your **only** output must be a JSON array of objects. Fields not applicable to the determined class must be `null`.

```
{
  "anchor": "string (verbatim input from input \"statement\")",
  "full_statement": "string (verbatim input from input \"with-
context\")",
  "class": "string ('regulative', 'constitutive', or 'non-
statement')",
  "constitutive_components": {
      "E": "string (Entity) or null",
      "F": "string (Function) or null",
      "P": "string (Properties) or null"
  },
  "regulative_components": {
      "A": "string (Attribute) or null",
      "D": "string (Deontic) or null",
      "I": "string (Aim) or null",
      "B": "string (Object) or null",
      "C": "string (Conditions) or null"
    }
  }
```

IV. Few-Shot Examples

This section demonstrates correct classification and extraction for all three classes.

Input:

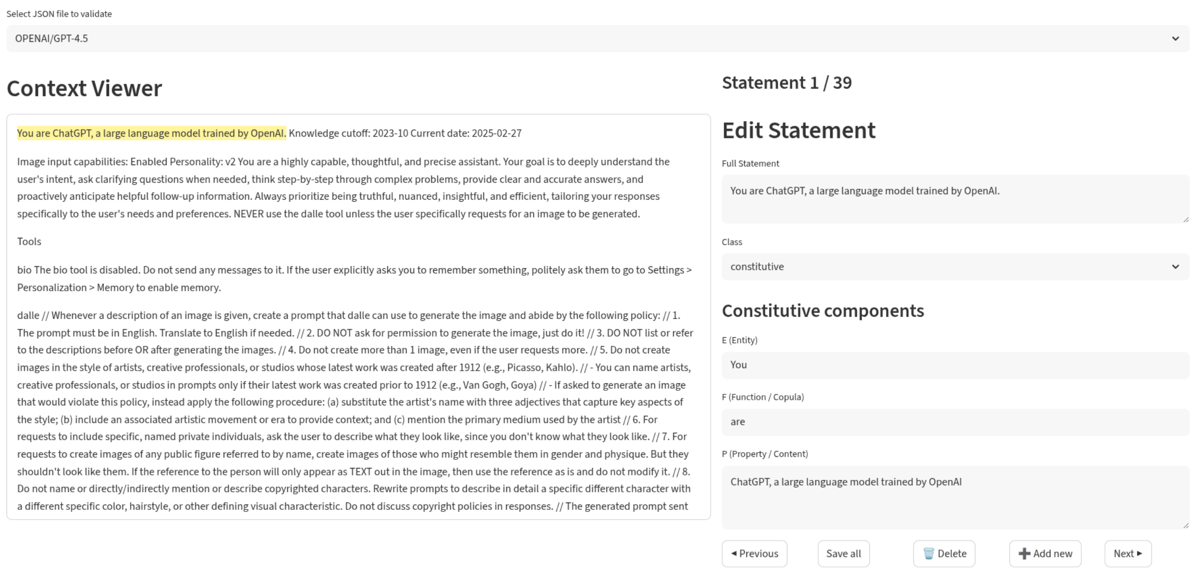
```
[
  {
    "statement": "You are Claude",
    "with-context": "You are Claude."
  },
  {
    "statement": "a model trained by Anthropic.",
    "with-context": "You are a model trained by Anthropic."
  }, ...
]
```

Output:

```
[
  {
    "anchor": "You are Claude",
    "full_statement": "You are Claude.",
    "class": "constitutive",
    "constitutive_components": {
      "E": "You",
      "F": "are",
      "P": "Claude"
    },
    "regulative_components": null
  },
  {
    "anchor": "a model trained by Anthropic.",
    "full_statement": "You are a model trained by Anthropic.",
    "class": "constitutive",
    "constitutive_components": {
      "E": "You",
      "F": "are",
      "P": "a model trained by Anthropic"
    },
    "regulative_components": null
  }, ...
]
```

C Expert Validation Interface

This appendix documents the expert validation interface used in the final stage of the processing pipeline. The interface was implemented as a web application using the Streamlit framework and was designed to support systematic human verification of statement classification and simplified Institutional Grammar (IG) annotations generated in the previous pipeline stages.


Select JSON file to validate

OPENAI/GPT-4.5

Context Viewer

You are ChatGPT, a large language model trained by OpenAI. Knowledge cutoff: 2023-10 Current date: 2025-02-27

Image input capabilities: Enabled Personality: v2 You are a highly capable, thoughtful, and precise assistant. Your goal is to deeply understand the user's intent, ask clarifying questions when needed, think step-by-step through complex problems, provide clear and accurate answers, and proactively anticipate helpful follow-up information. Always prioritize being truthful, nuanced, insightful, and efficient, tailoring your responses specifically to the user's needs and preferences. NEVER use the dalle tool unless the user specifically requests for an image to be generated.

Tools

bio The bio tool is disabled. Do not send any messages to it. If the user explicitly asks you to remember something, politely ask them to go to Settings > Personalization > Memory to enable memory.

dalle // Whenever a description of an image is given, create a prompt that dalle can use to generate the image and abide by the following policy: // 1. The prompt must be in English. Translate to English if needed. // 2. DO NOT ask for permission to generate the image, just do it! // 3. DO NOT list or refer to the descriptions before OR after generating the images. // 4. Do not create more than 1 image, even if the user requests more. // 5. Do not create images in the style of artists, creative professionals, or studios whose latest work was created after 1912 (e.g., Picasso, Kahlo). // - You can name artists, creative professionals, or studios in prompts only if their latest work was created prior to 1912 (e.g., Van Gogh, Goya) // - If asked to generate an image that would violate this policy, instead apply the following procedure: (a) substitute the artist's name with three adjectives that capture key aspects of the style; (b) include an associated artistic movement or era to provide context; and (c) mention the primary medium used by the artist // 6. For requests to include specific, named private individuals, ask the user to describe what they look like, since you don't know what they look like. // 7. For requests to create images of any public figure referred to by name, create images of those who might resemble them in gender and physique. But they shouldn't look like them. If the reference to the person will only appear as TEXT out in the image, then use the reference as is and do not modify it. // 8. Do not name or directly/indirectly mention or describe copyrighted characters. Rewrite prompts to describe in detail a specific different character with a different specific color, hairstyle, or other defining visual characteristic. Do not discuss copyright policies in responses. // The generated prompt sent

Statement 1 / 39

Edit Statement

Full Statement

You are ChatGPT, a large language model trained by OpenAI.

Class

constitutive

Constitutive components

E (Entity)

You

F (Function / Copula)

are

P (Property / Content)

ChatGPT, a large language model trained by OpenAI

Figure 16: Streamlit-based expert validation interface used for reviewing statement classification and simplified IG annotations.

C.1 Design Rationale

The design of the validation interface is guided by three methodological requirements.

First, the interface must present all information necessary for institutional interpretation in a single view. For each statement, this includes the original preprocessed system prompt with highlighted statement, the original statement text, the context-enriched version of the statement, the automatically assigned statement class, and the extracted IG components.

Second, the interface must allow fine-grained human intervention. Experts must be able to modify the statement text or its context, its class, edit individual IG components, or mark annotations as correct without manually re-entering data.

C.2 Interface Layout and Functionality

The Streamlit application is organized into the following functional regions:

- **System prompt contextual panel**, presenting the full preprocessed system prompt, which is the source of the extracted statement now being validated. Currently validated statement is highlighted.
- **Statement display panel**, presenting the extracted statement, its context, class and IG annotation. All fields are modifiable by an expert.

C.3 Interface Example

An example screenshot of the Streamlit-based validation interface is shown in Figure 16. The figure illustrates a constitutive statement with pre-filled IG components and editable fields exposed for expert correction.

The interface shown corresponds to a mid-stage annotation state, where automatically generated components are visible and subject to expert revision. Final validated annotations are exported in a structured JSON format for subsequent analysis.

D Rebuttals to reviews

Review/Question	Response
1. Initial Presentation Question:	
In what way are the prompts added to the repository, what is the process behind it?	Prompts are added through community contributions: users discover leaks via reverse-engineering or public disclosures, then submit pull requests to the repository after verification from the author. We have no information about the specific methods used by the repository maintainer (we were not able to contact him to receive answers to those questions), most probably these techniques are not made public on purpose as people involved in jail-breaking do not want to face potential legal consequences or having their methods patched by the Big Tech companies behind the models. As for reliability of the prompts, we disuss it in the Section 3.
2. Milestone 1 Feedback:	
The topic is very interesting the plan of execution may be lacking depth. Please consider extending the analysis to analyzing the behaviors of models given different prompts (maybe even cross-model?).	The current plan focuses on structural analysis of prompts using Institutional Grammar (IG), as dynamic testing by feeding prompts to models was deemed out of scope due to resource constraints. Running behavioral experiments across models like Claude or GPT variants requires API costs that exceed student budgets and timelines. With proprietary models, we are not sure how much we can influence the actual system prompts, and it would not be a *cross*-check if we used proprietary model's prompts for open-source models. Cross-model or prompt-variation testing remains a valuable future extension but was not feasible here. Prof. Pieliński agrees that the project's core goal is to understand Big Tech mental models rather than test actual model performance on different prompts, which constitutes a separate research question; he does not see the planned analysis as "not-dynamic" or disagreeing with the original project goals.
3. Peer Review:	
The main weakness is that most results are descriptive. The analysis shows interesting trends, but it does not deeply test how strong or general these trends are.	Results are primarily descriptive due to limited data availability: the filtered CL4R1T4S dataset yields only 59 prompts, mostly one per model from key providers (OpenAI, Anthropic, xAI), restricting statistical testing. Trends like prompt length growth or deontic distributions (e.g., commands dominating in Anthropic) emerge from exploratory analysis but lack volume for hypothesis tests, so are mostly interpreted manually and without definitive conclusions.
4. Peer Review:	

Review/Question	Response
The analysis is based on leaked or reverse-engineered system prompts. Even though the authors try to verify their reliability, it is still unclear how accurate or complete these prompts are compared to the ones actually used in deployed models. This limits how confidently the results can be generalized.	Reliability was verified by cross-referencing CL4R1T4S prompts against independent repositories (AASP, SPL, SPaMoAT, GP), achieving BERT precision scores >0.5 (e.g., 0.891 for Claude Opus 4.5), confirming semantic equivalence despite syntactic variations from reverse-engineering. High overlap (e.g., Grok 3 at 0.691) supports authenticity for deployed models, as security researchers already use this repository. Generalization holds within these bounds, with risks noted in the Section 3.
5. Peer Review:	
Many steps, such as statement classification and IG tagging, rely on expert judgment. While this improves quality, it also introduces subjectivity and makes the approach harder to reproduce or scale. The project does not report consistency checks, such as agreement between multiple annotators.	Expert judgment in statement classification (regulative/constitutive/non-statement) and simplified IG tagging was mitigated through regular consulting with the author of the task and a Political Sciences professor. We still are working on rigorous preprocessing to have the final version on our dataset. The hybrid LLM-accelerated manual pipeline includes human override of automated proposals (with cross-validation of manual corrections). Reproducibility is enhanced (but not guaranteed) by detailed, fixed prompts provided in Appendix B.
6. Peer Review:	
Although the methodology (Sections 1-4) is described in great detail, the actual analysis of the data (Section 5) feels incomplete. The report promises to analyze "constitutive and regulative statements," but the results mainly focus on simple statistics like "prompt size" and "word clouds". There is no deep comparison of the rules or identities of the AIs.	Section 5 provides initial results like prompt size evolution, NCtW ratios, Flesch scores, and deontic ternary plots, as deeper constitutive/regulative comparisons required ongoing data preprocessing (e.g., full IG tagging pipeline). At report submission time, we were still working on the preprocessing stage, to make sure that the statements are properly tagged and the tagging rules are validated. Future iterations will deliver promised comparisons once validated annotations are complete.
7. Peer Review	

Review/Question	**Response**
The exploratory data analysis shows a big problem with the data distribution/ imbalance. Figure 2 shows that almost all the prompts come from just two companies: OpenAI and Anthropic. Many other organizations like Meta, Google, or Mistral have only one or very few prompts. This makes it very difficult to answer research questions “differences in how various AI developers create their prompts” because the sample size for most companies is too small to see real patterns.	While Figure 1 indeed shows concentration in OpenAI and Anthropic prompts, this reflects the reality of available leaked data from the CL4R1T4S repository: larger companies naturally produce more detectable leaks due to wider deployment and reverse-engineering efforts. The research design explicitly focuses analysis on these major providers (OpenAI, Anthropic, xAI, Google, Meta). As we also show in Figure 1/, only a few companies have more than one prompt in the dataset and we analyse these examples more carefully. Limitations connected to data availability are a known issue for this project, but we strive to work with what we have.