# Are LLMs producing the sum of the web's parts? Structurally comparing model and human visual corpora

Hamidah Oderinwale* Eliana Gelman

*Core contributor

Generative models can produce individual artifacts that closely approximate web-made ones, but their differences at the level of individual generations may be too particular to those examples to make a generalizable hypothesis about their distinctiveness. We therefore study what separates the machine web and the web at the ecosystem level. We examine whether there are recurring properties across the generations of four frontier model families and a sample of the web, and whether those properties distinguish them. We sample real-world sites from the open web using Google's Chrome User Experience Report (CrUX) dataset (520k+ from 2016-2026 with a website that could be captured), and use these as references for synthetic briefs that LLMs produce generations from (500 being custom generated and 47k+ from the Web2Code dataset). This allows us to compare the 'web' and the machine web based on the same design intent, and to conduct this comparative analysis by looking at generative populations. We consider how the behavior of our structural findings change with sample sizes (with a matched set of generations to 500 web references and a fuller set of pages from the web, that is 800 times its size). We observe that models use much of the same vocabulary of elements as webs but compose those elements differently, producing a narrower set of recurring combinations. These structural properties distinguish web and machine artifacts and generations from different model families. We find that models fill more of their pages with compositions found in other generations, but the patterns they reuse are less deeply nested and rarely sample directly from the web reference corpus.

**Keywords**: Frequent pattern mining, model fingerprinting, visual generation, web homogenization

# Contents

**1 Introduction** **3**

**2 Why does this work matter?** **3**

**3 Summary of findings** **4**

**4 Methods** **5**
4.1 Using websites to prompt for briefs that models use for visual generations . . . . . 5
4.2 Frequent pattern mining . . . . . 5
4.3 Structural similarity by traversing DOM trees . . . . . 6
4.4 Ranking models by their population-level features . . . . . 7

**5 Background** **7**
5.1 How many generations does attribution need? . . . . . 7

**6 Sample efficient developer identification procedures: how much do you need to know?** **8**

**7 Variety** **9**

**8 What makes us think that language identification in the limit for the visual domain is actually possible?** **10**
8.1 Finding the right layer of abstractions: pixels and DOM elements . . . . . 10

**9 The structure of model generations** **10**
9.1 Model confusability and sample efficiency for attribution . . . . . 11
9.1.1 Modeling the relationship between confusability and identifiability. . . . . 11

**10 Predictability and canalization** **12**
10.1 Briefs to production . . . . . 12
10.1.1 Containability . . . . . 12
10.2 Compositional indicators . . . . . 13

**11 Discussion** **14**
11.1 The ecology of generations . . . . . 14
11.2 Related work . . . . . 14

**12 Conclusion** **15**

**A Appendix** **17**

**B Specifications of the generation process** **17**

**C Synthetic brief pipeline** **17**
C.1 Brief generation prompt . . . . . 17
C.2 Sample prompt . . . . . 18
C.3 Distribution of semantic categories . . . . . 19
C.4 Design brief analysis methods . . . . . 19

**D Comparison of visual encoders** **21**

# 1 Introduction

In this work, we try to understand the collective structure of visual artifacts on the web and of those generated by LLMs. We consider what we can glean when artifacts are assessed in groups, and what changes when they are observed through different lenses (e.g., visual embeddings and rendered DOM compositions). We also explore the space of design decisions that models *typically* make.

We are motivated by a number of questions. Firstly, as new models (and generative tools more broadly) become embedded in the web creative process how will the web change? Second, will we always be able to fingerprint the websites that models have contributed to and vice-versa? And third, what are the attributable design behaviors or features that make the web as we know it distinct? We find that models produce structurally distinct visual populations. Using a specialized implementation of frequent pattern mining, we characterize the recurring compositions that distinguish models from one another and models from the web.

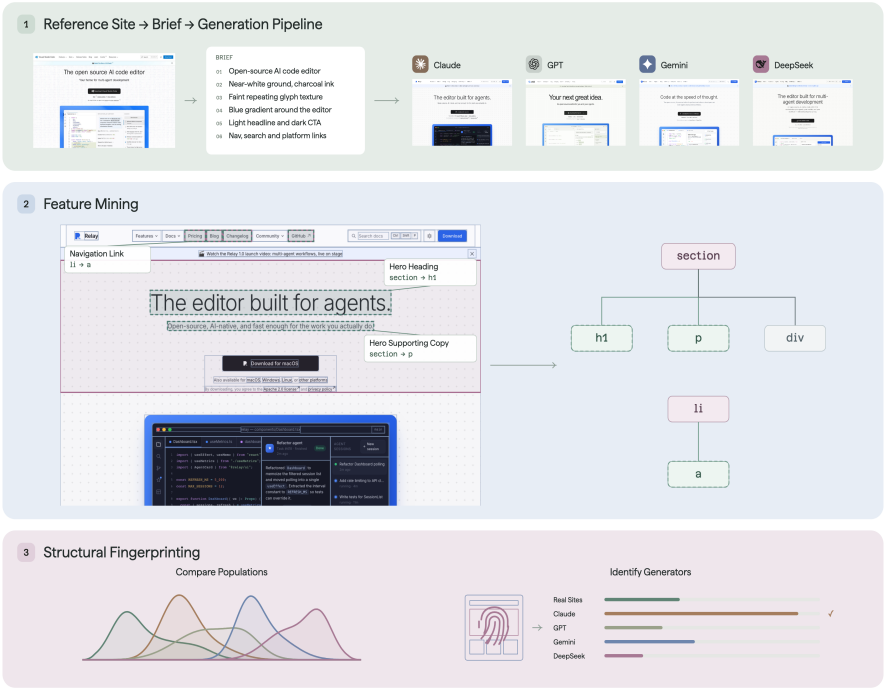


Figure 1: An overview of the pipeline of deriving synthetic briefs from reference websites, generating sites with the models, and extracting their recurring patterns and compositions.

# 2 Why does this work matter?

Given a population of generations, produced by LLMs trained on the entire internet—and a changing one at that—will models only ever approximate the web? Moreover, in a world where web-produced artifacts grows replete to train models, will outputs narrow on a small set of repeated patterns [1–3]? Production is shaped by the tools it employs, and design outputs are not exempt. When we consider how design outputs will evolve, we should consider the constraints on the processes that make them. Herbert Simon's framing of design as search under constraint provides an analogy for model sampling [4]. Extending this analogy, Waddington's concept of *canalization* offers a way to understand how restricted sampling can produce repetition: variation in the inputs gives way to a narrow range of generative outputs [5].

We find that models "model" the web in semi-predictable ways. The extent to which their unique styles can be forecast depends not only on the signatures they adopt, but also on how they approach their caricature-esque ways of production. Models become distinguishable from each other in the things they choose to emulate most strongly. This work is inspired by past work exploring the relationship between generation and identification and we set out to explore these ideas to the visual domain [6].

In this work, we try to make progress on:

1. Considering the abstractions that help us reason about macro-scale properties like diversity and redundancy that appear only in viewing the collective of individual generations [7]. The Web2Code dataset and the relevant subset of generations from GPT 3.5 give us insight into an older era of models as well [8].
2. Studying the global structure of the visual web through recognizable, web-interpretable, and traceable "features,"—learned through frequent pattern mining—in addition to finer-grained representations at the pixel and token-level. Do models and the natural web operate from the same semantics, and if not, how do they differ?
3. Understanding to what degree "mined" recurring elements can fingerprint generators.
4. A benchmark of representations that shows that structural features (recoverable through the DOM) offer more attribution signal than visual embeddings, while the text of a webpage offers the highest signal, where the order of signal is webpage text (82%) > structural features (71%) > visual embeddings (53% using SigLIP2 and an average of 52% across the four encoders we compare).[1]

# 3 Summary of findings

- The formation of compositions (the co-occurence of two or more elements in the same artifact) is a meaningful discriminator between the web and LLM generations. We define an element as the smallest unit in the DOM that can be rendered. There are recurring, co-occurring elements and compositions, in both LLM generations and on the web, that can identify them.
- Practically all (with a range of 99.6% to 99.8%) of the LLM generative populations are composed of observed patterns in the natural web.
- We consider a set of 500 model generations as a controlled group derived from the synthetic prompts based on human-made sites, and compare to the fuller dataset about 895 times its size. In it, 99.5% to 99.8% of a model's unique generative vocabulary, depending on the model, can be observed.
- LLM pages are less rich than the ones on the web. A given LLM web page has less unique recurring patterns or compositions (47.0 on average) than human sites, (64.1 on average).
- The average recurring composition from the web is also more nested than those from models. The average DOM element on the web has a nested structure of depth 10.65, and the average model has a nested structure of depth 5.65. This points to the element frequencies learned by each developer which picks up regularities but not the deeper structures the web naturally recycles.
- We rank developers by ecosystem-level metrics such as mean pairwise distance to see which ones track one another. **And more broadly, we observe that if you were to rank models by how many distinct elements their generations are composed of on average, this would not correspond to a leaderboard of structural diversity when considering the full corpus.**
- When considering the amount of information needed to do developer attribution, we consider the size of the compositions shown. We find that the highest efficiency attribution regime in our pattern-size comparison is one where a "classifier" is shown many smaller recurring patterns or compositions rather than larger ones. Smaller patterns also tend to recur on more pages, since matching a larger composition requires more elements to occur together in the same structure.
- Structural patterns are benchmarked against pixels represented by SigLIP2 embeddings, the text of the page, and the code style itself. We find that text is the most strongly attributable to the source (82%), followed by structural features (71%) and pixels (52%). We perform ablations to see the effect of different inputs for classification. Together, we find that the generating model can be recovered in a number of ways, all much higher than chance, with HTML providing the strongest additional signal.

[1] We benchmark four visual embedding models (SigLIP2 [9], DINOv2 [10], OpenCLIP ViT-L/14 and UIClip [11, 12]) and find similar values for all of them, with a mean of 52.0% for pairwise cosine similarity and a standard deviation of 3.6 (and a range of 48.2–56.7%) More information on the values can be located in the appendix in Table 4

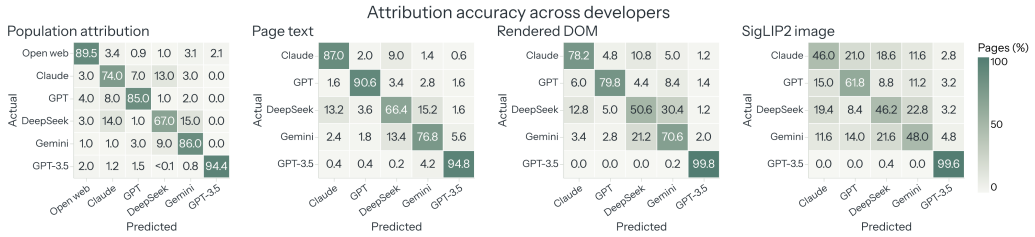


Figure 2: Fingerprinting signal across representations.

# 4 Methods

## 4.1 Using websites to prompt for briefs that models use for visual generations

To build our *web-reference corpus* we take a sample of 520k+ sites from the Google Chrome User Experience (CrUX) dataset. This is a snapshot of websites from the 2016 to 2026 (with an equal number drawn from five periods depending on when the pages were first observed: 2016, 2019, 2022, 2024, and 2026). *For the generated corpus*, we introduce a synthetic brief pipeline for producing prompts that are meant to emulate these sites (hence the specified length, where in a 10-site pilot we sweep across length ( 50, 200, 500, 750, and 900 words with observed median lengths of 96, 228, 548, 773, and 921 words) and temperature). We produce 500 briefs with four models (Claude Fable 5.1, GPT-6 Astra, DeepSeek V4 Pro, and Gemini 3.1 Pro) and evaluate our attribution methods on 100 of them as our hold-out group.

We find that longer briefs intuitively led to closer brand adherence to the original site, with temperature showing little effect. Brand adherence is calculated as the proximity to the web reference normalized by the visual distance to distant random references. We denote this as follows, where $\mathbf{e}_g$ represents the visual embedding of a model generation and $\mathbf{e}_h$ is the visual embedding of its web reference:

$$A(g, r) = \frac{\cos(\mathbf{e}_g, \mathbf{e}_h) - \mu_g}{\sigma_g}.$$

Given a larger pool of said briefs, we sampled sites to match the natural distribution of semantic categories (e.g., the Visual Studio Code website, `code.visualstudio.com`, in the SaaS & Software Product category). We also analyzed the briefs by their design themes. Across the 15 themes, the average share of briefs mentioning a theme was 95.8%, with a range of 79% to 100%. These briefs were designed to support "brand adherence" and had a target of 500 words, and were elicited by Fable 5.1. In practice, the range of lengths varies from 502 to 608 words, with an average of 545 words. More information on the briefs and a sample of a brief can be found in the appendix (see C).

## 4.2 Frequent pattern mining

To consider the space of visual generations structurally, we use frequent pattern mining (FPM) to identify recurring, nested patterns in the DOM[13–15]. FPM is a suite of techniques typically used for identifying recurring sets of items in a database. Here, we consider a database of pages, where each page is represented as a collection of DOM "compositions." While you could consider a more simple bag of DOM elements on their own, we also try and represent the structure of the elements themselves and how they compose, as this better represents the nature of a "motif"–symbols that are actually perceivable by humans, instead of an abstract ' ' with no content or styling. Trees can be used to model the global structure of these mined sequences. The DOM naturally has a tree structure, where every element can be tagged with its parent and its child. We consider the depth of these trees and model how nested compositions are depending on the developer. We count a pattern as recurring if it appears in at least two pages (minimum support). As such, we have frequencies within a developer population and across developer populations. Together, we observe 232 unique recurring patterns or *compositions*.

While we consider the hierarchy of elements within a composition, we ignore the order of sibling elements (elements that share a parent). We describe the algorithm for mining compositions in the DOM below:

| | Measure | Matched web (500) | Open web (528,801) | Claude (500) | GPT (500) | DeepSeek (500) | Gemini (500) | GPT-3.5* (47,862) |
|---|---|---|---|---|---|---|---|---|
| *Between corpora* | Overlap with matched web (%) | – | 29.8 | **40.6** | 36.4 | 36.5 | 33.0 | 34.9 |
| | Overlap with open web (%) | **29.8** | – | 22.7 | 18.8 | 18.1 | 14.5 | 18.9 |
| | Own compositions found in open web, mean of depths 1–4 (%) | 91.0 | **100.0** | 77.5 | 70.9 | 73.9 | 77.6 | – |
| *Within corpora* | Distinct compositions | 10,567 | **35,403** | 8,076 | 6,657 | 6,456 | 5,149 | 6,707 |
| | Mean distinct appearance labels per page | **64.1** | 54.9 | 60.3 | 48.8 | 43.4 | 35.6 | 10.8 |
| | Mean DOM element depth | **10.65** | 10.25 | 6.28 | 5.56 | 5.45 | 5.31 | 3.05 |
| | Average depth of a recurring composition | 2.78 | – | 2.69 | 2.48 | 2.18 | 2.19 | **3.01** |
| | Largest depth of a recurring composition | **16** | – | 9 | 8 | 8 | 8 | 7 |

* Web2Code column uses renderable, valid pages attributed to GPT-3.5 (47,862 unique assets). Shading corresponds to the magnitude of a value. Bolded entries are first ranked for that metric. Unless stated otherwise, each measure corresponds to a recurring pattern of depth 1. Developers share a rank when their metrics have the same value. **Findings:** Among models, Claude has the highest average position across the selected measures, followed by DeepSeek, GPT, and Gemini. DeepSeek has the largest corpus-wide set of distinct elements among models.

Table 1: Ecosystem-level comparisons across developers.

**Mining recurring DOM compositions**

**Require:** For a set of pages grouped by its population and its DOM elements.
composition depths $H \subseteq \{1, 2, \ldots\}$

**for all** producers $r$ **do**
  Set the page count $n_r(c) \leftarrow 0$ for every composition $c$
  **for all** pages $p$ of producer $r$ **do**
    For each element of $p$, find the element it is nested inside
    $C_p \leftarrow$ the distinct compositions on $p$ at depths $H$
    **for all** compositions $c \in C_p$ **do**
      $n_r(c) \leftarrow n_r(c) + 1$
    **end for**
  **end for**
  $\mathcal{R}_r \leftarrow \{c : n_r(c) \geq 2\}$
**end for**
**return** recurring compositions $\mathcal{R}_r$ and page counts $n_r$

## 4.3 Structural similarity by traversing DOM trees

With the structure that we then have, we use Jaccard similarity over these sets of patterns between pages and take the sum of these pairwise page **structural similarities** for our analyses. We compare sets of recurring patterns of a depth of 1 (indexed by 0). There is also a variant where we consider the longest chain, where we find the recurring patterns of longest depth where there is another that also exists, $J(A, B) = |A \cap B|/|A \cup B|$.

Consider the following depth-2 combinations:

$$A = \left\{ \begin{array}{l} \texttt{header} \rightarrow \texttt{navigation} \rightarrow \texttt{link}, \\ \texttt{main} \rightarrow \texttt{card} \rightarrow \texttt{image}, \\ \texttt{main} \rightarrow \texttt{card} \rightarrow \texttt{button} \end{array} \right\}, \qquad B = \left\{ \begin{array}{l} \texttt{header} \rightarrow \texttt{navigation} \rightarrow \texttt{link}, \\ \texttt{main} \rightarrow \texttt{card} \rightarrow \texttt{image}, \\ \texttt{main} \rightarrow \texttt{section} \rightarrow \texttt{heading} \end{array} \right\}.$$

While the measure alone does not account for frequency, we can apply a simple recurrence parameter to take it into account. We can denote it as follows:

$$J_w(A, B) = \frac{\sum_p \min(g_A(p), g_B(p))}{\sum_p \max(g_A(p), g_B(p))},$$

where $g_A(p)$ and $g_B(p)$ are the number of appearances of pattern $p$ in each corpus.

### 4.4 Ranking models by their population-level features

In this work, we focus on ecosystem-level characteristics of the web artifacts and model generations. For a given metric, we consider the sum of the pairwise distances of our metrics (e.g. structural similarity or cosine similarity from embeddings) [7].

## 5 Background

Previous work has looked into language identification in the limit, which shows that a generation can always be attributed to a set of rules for some language given an accumulating number of samples, where the test is whether given those rules a new generation can be pointed back to the learner of said rules. Here, we consider how collective samples can be used in the visual domain, we make the further claim that in this domain where the exploration space is infinite, collective frequencies are paramount for attribution. The work also posits that in theory it would be possible to attribute a models unseen generations given a sample by learning its languages which can be done to varying levels of efficiency given its structural proximity to known model families.

### 5.1 How many generations does attribution need?

Given the similarity between two model families, measured by the mean of their pairwise distances, we find that more "related" families require more data to distinguish. For every point increase in structural similarity–measured by another shared element across the two model corpora–accurate attribution requires 7.8% more information than if the two theoretical corpora shared one less element.

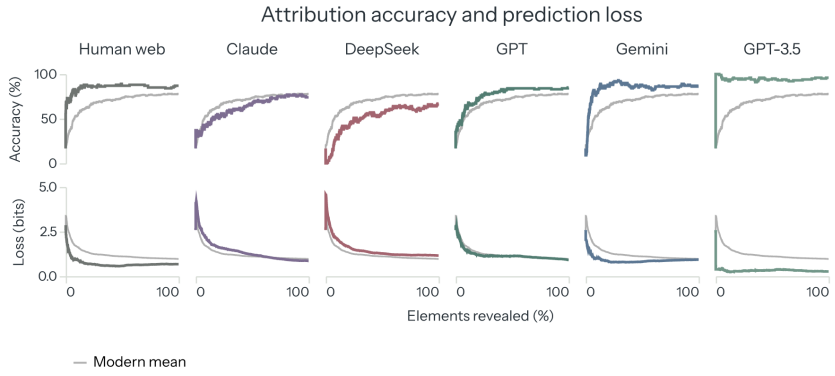


Figure 3: Prediction accuracy by amount of information (elements)

We consider the frequency of 232 unique elements across generations. Pages from the web saturate 19.6% on a matched set and models saturate 9.6% to 15.0%. [2]. Given eight generations (artifacts) from the same model, we identify what model produced the samples with 99.6% accuracy among four known model families. GPT 3.5, a dataset of generations that did not go through the pipeline surprisngly had the highest attribution accuracy (99.8% of its generations were correctly classified and the one wrongly attributed one to Gemini).

We compare model families by the overlap in their element-frequency distributions. Across the six model pairs, greater similarity was associated with more generations needed to distinguish them. The closest pair (Gemini and DeepSeek) had 80.0% overlap and required three generations to reach 95% attribution accuracy, compared to two generations for the most distant pair (GPT and DeepSeek), with 65.9% overlap.

"Model collapse" describes an observed phenomenon in which models recursively trained on generated outputs lose information about the original distribution [1]: models forget the ends of the distribution, so rarely represented facets of a pre-training corpus are lost over time. In our matched comparison, where web references supply the briefs that models generate from, we see that models are prolific repeaters of compositions compared to the web. We hope this can inform future work on what models would produce and how you can train for corpus-level measures like diversity.

[2]With a theoretical limit of $232^2 = 53{,}824$ combinations

This makes sense, since diversity and mode collapse are measured at the set level and are difficult to train for. For a given composition, a model explores ∼90% of a web's variations; however, the space of compositions a model draws on is much smaller, and the ones that appear are the most popular web ones.

The Platonic Representation Hypothesis argues that models converge in their representations of reality [16]. We consider whether independently developed models also produce converging outputs.

# 6 Sample efficient developer identification procedures: how much do you need to know?

To measure how identification changes as more elements are revealed, and how this differs across developers, we employ the following procedure. We employ depth-first traversal: starting at the roots of the rendered element tree, we reveal each parent before its children, following their spatial order. This shows an accuracy of 79.8%. [3]

**Attribution procedure from revealed elements**

$s_d \leftarrow 0$ for each developer $d$
**for** $t = 1, \ldots, n$ **do**
    $s_d \leftarrow s_d + \log P(k_t \mid d)$ for each developer $d$
    $\hat{d}_t \leftarrow \arg\max_d s_d$
**end for**

After each element is revealed, a classifier using naïve Bayes predicts the page's source from its growing set of elements. We use the frequency-labeled elements by developer learned from the "training set."

We also consider the ranked similarity metric, using family similarity to examine the next-best prediction in the list. **We find that most attributions are picked correctly on the first or second try.** We denote the classifier as a function of revealing elements of given generations and the classifier's identification of a developer as, where the goal is to select the developer of highest likelihood given observations.

The classifier operates by computing the following, where $D$ is the developer and $G_d$ represents the set of generations for a given developer, $g$ indexes a generation, and $n$ represents the number of elements revealed in the sequence:

$$P(D_g = d \mid K_{g,1}, \ldots, K_{g,n}) \propto P(D_g = d) \prod_{t=1}^{n} P(K_{g,t} \mid D_g = d).$$

Investigating the attribution efficiency process, we consider how much information is needed to perform attribution well. We find that if one considers every element in terms of its order, and then see how much more information is learned with each recurring smallest recurring pattern.

[3] We find that the similarity of alternatives correlates with confusability at a ranked agreement of 0.52 using Spearman's correlation.

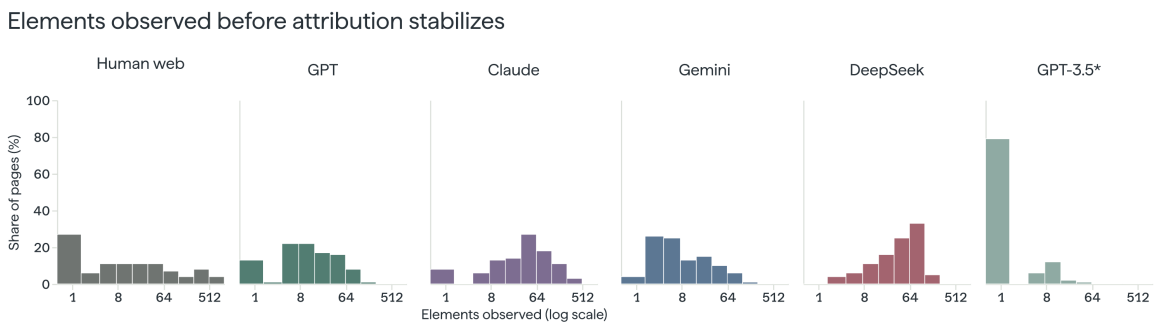


Figure 4: Number of elements observed before the identification of the source stops changing on a holdout dataset (100 generations). Gemini is a model for whom the first elements shown are the most crucial both in terms of the order of information on a given page and elements in the corpus more broadly, where the marginal element is of diminishing value in the most extreme way, followed by the web and GPT. DeepSeek is relatively back-loaded where the number of elements required for attribution is at the 33rd percentile of elements on the page. Claude has the most evenly spread distribution.

Across models, 32.3 (38.4 including inaccurate predictions) elements is the average number shown before the ability to identify who developed the marginal generation stabilizes, where after that number of elements while for the average web reference it is 44.9 elements (77.3 including inaccurate predictions). We define these numbers by considering the element of the n'th order, where no additional element being added to the context for fingerprinting improves the accuracy of the attribution. The standard deviation across pages is larger for web references, which makes sense given their visual variance (179.9 elements compared to 47.5 for model generations). The average number of elements shown before identification stabilizes has a standard deviation of 22.7 elements across models. The distribution across pages is right-skewed, which means that some pages require many more elements to be shown before identification stops changing.

Overall, we find that models are unique in what they produce compared to the web. We question whether at the population-level there are patterns in their development that structurally differ from the web. Given that they are in fact distinct, we then consider if we can model dynamics at scale.

We find that every model's generations are closer to model generations than any web artifact, where models can be distinguished from one another with 82% accuracy from rendered structure alone (91% when pixels and text are read with it) and web artifacts can be told apart from machine generations with 97% to 98.5% accuracy. Given a set of 500 synthetically generated sites from a pipeline that builds sites from web references, we are able to identify generations as machine-made with 99% recall.

In our table 1, we show what models have the highest overlap with the web. We can also see what models most overlap with each other. 68% of the compositions in Gemini generations are also found in Claude generations. But only 17% of Claude recurring patterns are seen in Gemini generations. This analysis gives us a rough proxy for the breadth of these producer vocabularies and the marginal visual contributions of models to the generative population (see more in Table 5).

# 7 Variety

To understand variety, we employ the FPM methodology. We observe 232 learned individual elements, meaning the total number of possible compositions is 53,824 ($n^2 = 53{,}824$). We find that natural web pages contain 10,387 distinct two-element compositions whereas the LLMs average 6,585.

Given this set of unique recurring elements, we find that models compose them into unique recurring patterns (compositions) at a smaller rate of 17%–34% less than the observed web corpus.

Additionally, comparing the Web2Code dataset of GPT 3.5 generations released in April 2023 against the rest of the model output corpus and CrUX dataset provides a snapshot of model output change over time. We observe that it has the least amount of distinct recurring compositions with 48.3% less than average than the 'matched' web corpus, 82.2% than the full web corpus, and 42.5% less than the next lowest model (Gemini). Additionally while the range of each developer's vocabulary size (including the web's) ranges from 214 to 227

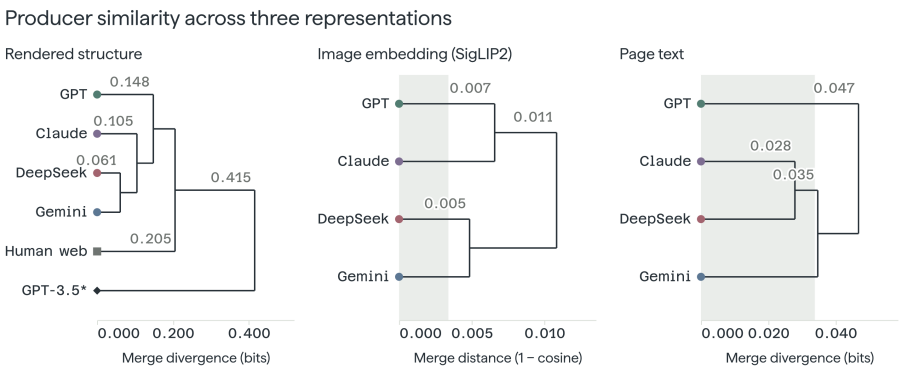


Figure 5: The recoverability of the model identity by representation. The similarity of models by their rendered structural elements. We find that all the models cluster by structural similarity more closely to other models than to the web corpus. Claude, however, is the closest to the web, with 32.0% overlap in structural combinations, compared to 38.1% with GPT, its closest model, based on Jaccard similarity.

[4], GPT 3.5's is 192. One might presume this is largely a function of the smaller training corpus that the model learned on (as a function of a smaller web at the time of the model's release in April 2023.

Across models, a given page has about 17 less unique recurring compositions on average than a web reference page (47.0 (SD = 17.2) versus 64.1 (SD = 22.1). Across the collection, the web has 224 unique patterns, compared to a model average of 216 (SD = 1.4).

The average length of the models' recurring compositions is also longer (by an average of 1.5 more elements) than the web. Additionally, more of their compositions are the same element repeated side by side (runs 0.8 elements longer on average than the web), which suggests that models 'copy' through learned behavior while the web's sites repeat, albeit less frequently. There are 3,177 compositions (two co-occurring elements) that exist only in the web corpus, compared to a random baseline of excluded compositions being about 970.

# 8 What makes us think that language identification in the limit for the visual domain is actually possible?

## 8.1 Finding the right layer of abstractions: pixels and DOM elements

The similarity of an LLM site to a site on the web is as close in visual pixel similarity (range of 0.757–0.774 cosine similarity) as any web site is to another (0.774 cosine similarity). Thus, given just pixels the natural web and the machine web is hard to discriminate. However, this motivates our focus on structural similarity.

We find that pixel and structural (rendered DOM elements observed via image) measures agree with 54% pairwise agreement for model generations but direct source code and pixels do not (see Figure 5).

Code nevertheless provides useful structural information for design representation. In order for code to render, it generally needs to compile or otherwise be parsed and executed; compiled design programs are therefore syntactically validated. Syntax trees and DOM structure provide hierarchy and *order* that are unavailable in pixels alone. Finally, code lets us distinguish observed artifacts from the broader space of syntactically correct but unobserved programs that could be generated and rendered.

# 9 The structure of model generations

We hypothesize that if historically models are trained on web content they will be hyper-optimized to emulate web artifacts but due to Goodhart's law, will produce caricatures of the training data samples they are exposed to. The existence and trends among these caricatures are the basis of the fingerprints we have been able to build of different models. From them, we learn that generations across a variety of models cluster together in ways that not only us to distinguish models from each other but also from web references to an even higher

[4]The matched web corpus' vocabulary size is 224 and the extended corpus' vocabulary size is 227.

degree of accuracy (94.2 % model-web average accuracy with a standard deviation of 1.1, compared to 89.4% distinguishability or pairwise attribution accuracy rate of 89.4% with a standard deviation of 6.3). See 1 for the full list of attribution accuracy rates on a hold-out.

We also consider the Web2Code dataset, of which we study 47k generations from GPT 3.5.[5] The generations from this older model give us an understanding of whether our findings persist for generations that were not produced from our pipeline.

Firstly, we rank models based on the density of the recurring patterns they share with the web corpus Claude (32%) > Gemini (31.9%) > DeepSeek (31.3%) > GPT (30.5%).

## 9.1 Model confusability and sample efficiency for attribution

First, we consider the degree to which feature overlap causes "confusability" (where a generation is wrongly attributed to a source). It would make sense that if two developers are confused with another often, that they would share a larger set of recurring patterns than other developers. We find this to be true, where more confused model pairs require more generations to be correctly identified on a holdout set, where the rankings of similarity and attribution largely agree with each other (Spearmans's agreement of 85% across six pairs, defined by $\binom{4}{2} = 6$. Namely, the more similar a pair of developers by comparing the structural similarity of a set of their generations from the same reference, the more generations needed to saturate their accuracy (i.e. at a rate of 95% where the marginal return of another generation in the sample decreases).

To understand this in practice, we come up with an ordering of models by their confusability. Confusability is an assymetric property of model generations from distinct developers, where a model B being falsely attributed to model A as its developer, does not mean that a generation from model B being falsely attributed to model A will occur at the same rate (see Figure 6 as a visual example).

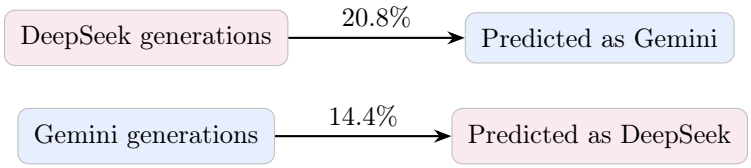


Figure 6: The directionality of misattribution.

That said, Gemini being predicted as DeepSeek 14.4% of the time and DeepSeek being predicted as Gemini 20.8% of the time. We list the rest of the rates of confused pairs in the diagrams below, out of pairs in a given order, the average rate is 7.7%. And when we consider the sample efficiency for attribution of Gemini generations, it requires twice the number of samples to be correctly distinguished from DeepSeek models compared to Claude generations (the second most frequent confused developer of Gemini generations), where the number of samples needed are four and two, respectively.

### 9.1.1 Modeling the relationship between confusability and identifiability.

Together, we find that while model famillies have distinguishable populations, they are not equally distinguishable. Thus, we try and quantify the magnitude of distinguishability as a function of pairwise confusability and the rate of identifiability. We plot the confusable pairs and the learning curves below to show the amount of samples needed below.

Given pairs of generations from the same source that are increasingly confusable as a function of their distinguishability from each other relative to every other generation from other developers, sample efficiency can be measured by how many more generations you need to reach to get to the same rate of accuracy given that the pair of developers was of baseline confusability.

Gemini and DeepSeek are the most confusable pair, with Gemini being predicted as DeepSeek 14.4% of the time and DeepSeek being predicted as Gemini 20.8% of the time. Next is DeepSeek and Claude, with DeepSeek being predicted as Claude 16.1% of the time and Claude being predicted as DeepSeek 7.2% of the time. We list the confusion rates for the remaining pairs in the diagram below; across all six pairs, the average two-way confusion rate is 7.7%.

[5]Web2Code is comprised of 800k+ pages, only a subset are attached to a model generation (a total of 60k–we filter for those that meet our rendering checks)

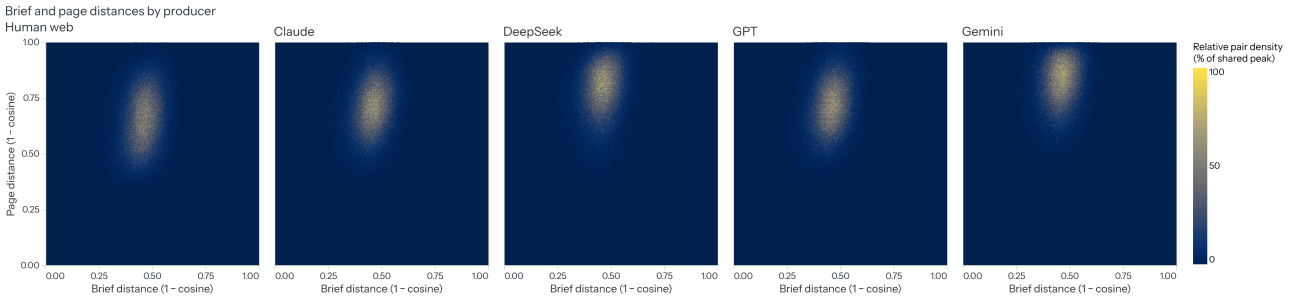


Figure 7: The distribution of the brief-page similarity. We see that there is a weak positive correlation between the variance of the briefs and the variance of the pages in pixel space. You can observe the light region of each map being concentrated on the x-axis and changing position primarily on the y-axis. This suggests that most page-brief pairs are relatively concentrated, but given the distance of two briefs, the pages they produce will be relatively more distant.

# 10 Predictability and canalization

## 10.1 Briefs to production

We study the predictability of a site, where given the intent, we consider the most likely set of compositions that are likely to appear. We find that model generations are much higher entropy than web sites when stratifying by intent where it takes 5.60 bits of information to predict the next element compared to 4.43 for web sites. Additionally, from the same grammar, the models are more likely to produce a wider range of things.

To understand the relationship between briefs and design outputs at the population level. We embed the briefs and compare distances between every corresponding pair of generations relative to them (for a total of 124,750 pairs per corpus, $\binom{500}{2}$) using `Qwen3-Embedding-0.6B`. We take cosine similarity across embedding space and TF-IDF for our structural element space.[6] We find that briefs very loosely but consistently track with page structure, where their orders have a rank agreement of 0.16–0.24 across models. The distance between any given two pages from the same corpus is similar for both LLMs and the web with a grouped mean distance of 0.95. Generations from the same brief average a distance of 0.90 while those with distinct briefs average 0.95.

We see that there is 97.9% accuracy in terms of correct classification of a site when you use structural features instead of comparing pixels which provides 79.0% accuracy. This suggests that the how of production is of signal itself and not only the granularity of how it is done. We see that the models generate artifacts that are narrower than the web though. We measure this by looking at how many combinations the developers explore compare to what is possible. The share of primitive elements are majorly overlapping (only differing by 3%, with models covering 95% of the space compared to 98% of models). But when we measure the average length of recurring elements, the models produce longer ones, despite producing a smaller library overall that is 3,788 compositions smaller on average (models have an average library size of 6,599 distinct compositions and the websites have a library size of 10,387), while the models' recurring compositions are 1.5 elements longer on average (models having 6.25 vs. the web with 4.89 elements).

Given some grammar, we also set out to understand how much different generators *explore*. We see that webs explore more than models when considering the variations of the structural patterns they produce, but are much more predictable than models in terms of entropy. For entropy, we consider for some element on the page, the amount of information that is required to predict the next items in order. We find that it takes 4.2 bits of information to predict the next variation of some structure in a web page compared to 4.6 to 5.2 for models. The size of pages by developer also differ with web pages begin the largest token-wise.

### 10.1.1 Containability

We see if the model is identifiable, how related is this to distinctiveness and future generations and its ability to make a new contribution (adding structures in newly placed formations from the broader vocabulary). We first ask how much each of the common recurring smallest elements co-occur. We find that DeepSeek is the most containable in Gemini at a rate of 63.6%, with an average "containability" of 54.5% across all pairs. We

[6] weighing every parent-child element combination

also consider how often these elements appear which gives us an understanding of the distribution. When we stratify by the intent and see the agreement across the common co-occurring elements for every model, we see that there is strong agreement as well of 94%.

We consider this in a theoretical case to model the ecological implications of this behavior. In theory, models, given a smaller toolkit, the webs and models become more indistinguishable. It is in investigating combinatorial depth that you see difference and developers become distinguishable. With each element, identifiability drops by 75% on average across models.

We can measure the loss in identifiability as a function of information in elements as:

$$C_{\text{geo}}(k) = \left( \prod_{m=1}^{4} C_m(k) \right)^{1/4}.$$

$$C_m(k) = \frac{\text{web combinations of length } k \text{ also seen in model } m}{\text{Total web combinations of length } k}. \quad (1)$$

We calculate this as the geometric mean where there are four models and we consider all the factors in tandem, resulting in the following:

We consider for an element type, how many options are effectively open for the following element. For the most common structural patterns—where a structural pattern is considered from the DOM Tree and is an element than can be nested and hava child element such as a div—, we see the number of variations for webs (where there is always more, 83 variations per type) and then rank the models accordingly, where Claude is the least predictable with an average of 58 variations per type, then GPT (49), DeepSeek (47), and Gemini (44). Across all structural types and variations, models explore 78% of the web's exploration space which encompasses 30% of the total. We see that the loss of variety is the same on common and rare structural patterns.

We normalize by the amount of content on the pages as well because web sites tend to have 5.9 times more content when measured in tokens with the web median of 5,900 compared to the model median of an average of 1000. They are also more nested, where web pages average 10.6 levels of depth, compared to 5.65 for model pages (SD = 0.44 across model means).

## 10.2 Compositional indicators

Beyond just single elements alone, we look at the specific compositions that models build by using the natural nesting of the DOM and co-occurence from our FPM methodology. We recover the elements from the rendered pages which allows us to use this approach even without having access to the source code. At the composition-level, models become particularly finger-printable. Almost all of the features are related to something navigation-wise or structural to the page. Interestingly these are facets of sites that are difficult for webs to code but models build their own unique patterns and re-use them often. For Claude and Gemini they are nav bar related compositions, for DeepSeek it is a gradient hero and for GPT it is a box with a glyph inside.

We find that most of the frequent re-curring compositions (meaning it appears at least two times in a corpus; in two generations for the same model or across a generation for two or more models) are of size two to five—these are small "motifs" that are hard to discern. This suggests that AI-generation tells are not necessarily distinctive in isolation; rather, their combination can produce a highly. At larger composition sizes, a model becomes easier to identify 75% of pages on average across sizes, with the average reused composition being six elements), although the number of recurring patterns decreases. Some models exhibit higher frequencies of smaller compositions, whereas others exhibit more frequent larger compositions. We find that the frequency of a composition, more than its size, often yields the greatest identifiability 70% of pages on average across the four models. DeepSeek has the highest amount of compositional reuse at 62%, much higher than the average (30%), followed by Claude (28%), Gemini (23%) and GPT (8%), against 12% for the web.

We find that models learn individual elements but they repeat certain combinations at higher rates. Logits are used to model next-token probabilities, but combinations give us a interpretable level of abstraction to consider more meaningful elements in visual design space [17, 18]. The size of these patterns help with making objects recognizable but on paper do not change how much they vary. This shows a gap between web perception of distinctiveness and surface-level measures. We recognize the limit of this approach which is the minimum support needed for a pattern, we are not aware of how many generations you need to see a given combination

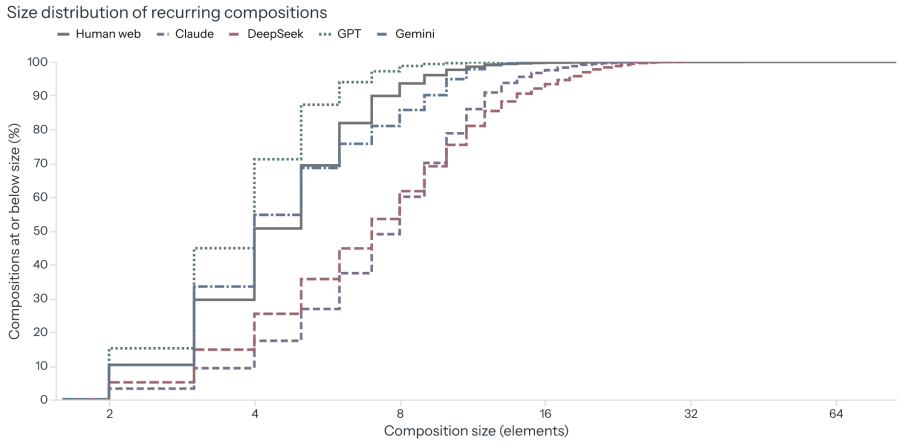


Figure 8: Cumulative share of reused compositions by size. Claude and DeepSeek have larger recurring compositions than the general web on average. Every curve represents the share of recurring compositions as compared to the distinct recurring compositions by a developer.

again. The smallest recurring compositions (two elements) appear on a median of 5% of generated pages. Across composition sizes two through five, median prevalence decreases by 26% on average per additional element.

The mined compositions appear on at least seven of the 500 generated pages per model, although their observed frequency varies with composition size.

Given this toolkit for studying model populations, we're also capable of benchmarking them against each other by their overlap with the patterns that webs produce and the ones the degree to which they overlap with the web. While all models are structurally distinguishable from the web and can be ranked on their distance, the numbers are consistent.

# 11 Discussion

## 11.1 The ecology of generations

As preference models are built to judge the quality of LLM artifacts and we explore learned reward signals to train models in iterative improvement loops, the directions in which they steer generation remain difficult to interpret. Future work might explore the learned preferences of these models with interpretability techniques. Additionally, we explore the minimum set of candidate generations needed to fingerprint a model. Future work could tie these findings back to model lineage and examine whether this minimum set changes depending on the set of known models. This could help characterize the distinctiveness of a new model and, alongside measures of quality or utility, its marginal contribution to the generator ecosystem. Lastly, we consider a reference as the baseline for generations and study model use in what is arguably a vacuum. Future work might explore how these structural fingerprints change when models are used with the tools that developers employ in the wild (e.g., harnesses) and consider the limitations of our prompt generation pipeline and the noise that it might have introduced. In the future, one could consider using prompt optimization frameworks like DsPy to make sure that specifications are programmed and not only instructed. In future work, we may also consider the population-level features of different models in the same family, to also understand if there may be evolutionary traits that can be observed as well [19].

## 11.2 Related work

**Training interventions.** Can diversity or richness be trained? Some would assume that past observations, references, and knowledge are required to push the boundaries of what can be generated. However, there is a growing body of literature aiming to study the degree to which the above is true and what can be post-trained into a model. [20] West et al. claim that base models are more creative than specialized models in game-like environments and poetry writing. The DARLING framework considers post-training interventions by sampling

and 'verbalized sampling' is introduced in recent work by Zhang et al. to prompt models to give responses matching a diverse distribution while keeping commonsense [21].

**Studying populations.** This work draws from a number of schools of thought. Ecologists and demographers have made much progress studying populations and their dynamics [22]. The Hill numbers encompass measures of richness, diversity, and entropy within ecosystems that map to different levels of organization. Mode collapse is an observed failure mode in LLMs.

**Fingerprinting.** In terms of fingerprinting, code stylometry has been explored to see what quirks of webs persist in the code they write [23] and in multimodal models more broadly by looking inwards at their internals [24]. We take inspiration from those who have previously used pattern mining over the DOM.

**Web dynamics.** Ruth et al.'s work offers a study of the web from a user perspective, namely how much time people spend on various websites drawing from the Google CrUX dataset and the popularity of websites in the dataset. Motifs are a term native to a number of fields including literature and are used to describe recurring themes and symbols. Chou uses them to derive a formal grammar of stories and to study their networked structure [25].

## 12 Conclusion

In this paper, we focus on characterizing the population-level features of LLM generations and web artifacts, and we find that they are structurally distinct. We find that pages from the web have more deeply nested compositions than the average LLM-generated page. Models produce long generations with substantial content but shorter recurring patterns in terms of depth or "nesting." Future work could examine whether these differences reflect how the web and models reuse existing compositions rather than construct them from scratch.

## References


[1] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, July 2024.

[2] Jon Kleinberg and Manish Raghavan. Algorithmic monoculture and social welfare. *Proceedings of the National Academy of Sciences*, 118(22):e2018340118, 2021. doi: 10.1073/pnas.2018340118.

[3] Manish Raghavan. Competition and diversity in generative ai. *Management Science*, October 2026. ISSN 1526-5501. doi: 10.1287/mnsc.2025.02761. URL http://dx.doi.org/10.1287/mnsc.2025.02761.

[4] Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.

[5] C. H. Waddington. Canalization of development and the inheritance of acquired characters. *Nature*, 150: 563–565, 1942. doi: 10.1038/150563a0.

[6] Jon Kleinberg and Sendhil Mullainathan. Language generation in the limit, 2024. URL https://arxiv.org/abs/2404.06757.

[7] Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations, 2025. URL https://arxiv.org/abs/2509.02534.

[8] Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, Timothy Baldwin, Zhengzhong Liu, Eric P. Xing, Xiaodan Liang, and Zhiqiang Shen. Web2Code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.20098.

[9] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features, 2025. URL https://arxiv.org/abs/2502.14786.

[10] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

[11] Jason Wu, Yi-Hao Peng, Amanda Li, Amanda Swearngin, Jeffrey P. Bigham, and Jeffrey Nichols. Uiclip: A data-driven model for assessing user interface design, 2024. URL https://arxiv.org/abs/2404.12500.

[12] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 2818–2829. IEEE, 2023. doi: 10.1109/cvpr52729.2023.00276. URL http://dx.doi.org/10.1109/CVPR52729.2023.00276.

[13] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15:55–86, 2007. doi: 10.1007/s10618-006-0059-1. URL https://doi.org/10.1007/s10618-006-0059-1.

[14] Tapan Parikh. Forty questions: Identifying a language model from one-word answers, 2026. URL https://arxiv.org/abs/2610.28281.

[15] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, pages 207–216, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897915925. doi: 10.1145/170035.170072. URL https://doi.org/10.1145/170035.170072.

[16] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis, 2024. URL https://arxiv.org/abs/2405.07987.

[17] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[18] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1675–1684. Association for Computing Machinery, 2016. doi: 10.1145/2939672.2939874. URL https://doi.org/10.1145/2939672.2939874.

[19] Benjamin Laufer, Hamidah Oderinwale, and Jon Kleinberg. Anatomy of a machine learning ecosystem: 2 million models on hugging face, 2025. URL https://arxiv.org/abs/2508.06811.

[20] Peter West and Christopher Potts. Base models beat aligned models at randomness and creativity, 2025. URL https://arxiv.org/abs/2505.00047.

[21] Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock llm diversity, 2026. URL https://arxiv.org/abs/2510.01171.

[22] M. O. Hill. Diversity and evenness: A unifying notation and its consequences. *Ecology*, 54(2):427–432, 1973. doi: https://doi.org/10.2307/1934352. URL https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/1934352.

[23] Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss, Fabian Yamaguchi, and Rachel Greenstadt. De-anonymizing programmers via code stylometry. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 255–270, Washington, D.C., August 2015. USENIX Association. ISBN 978-1-939133-11-3. URL https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/caliskan-islam.

[24] Chao Huang, Meng Tong, and Kejiang Chen. Fingerprinting multimodal large language models, 2026. URL https://arxiv.org/abs/2609.20457.

[25] Jean-Peïc Chou and Marc Christie. Structures in tropes networks: Toward a formal story grammar. In *Proceedings of the Twelfth International Conference on Computational Creativity*, pages 32–41, Mexico City, Mexico, September 2021. URL https://inria.hal.science/hal-03777738v1.

[26] Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, Michael Stern, Brian Clarke, Landon Goldberg, Theodore R. Sumers, Jared Mueller, William McEachen, Wes Mitchell, Shan Carter, Jack Clark, Jared Kaplan, and Deep Ganguli. Clio: Privacy-preserving insights into real-world ai use, 2024. URL https://arxiv.org/abs/2412.13678.

# A Appendix

# B Specifications of the generation process

We run generations in a custom harness. Every generation employed models that had access to file-editing and shell tools with a maximum of 50 steps, with reasoning enabled in an offline container. The default temperatures for all of the models were used (Claude = provider default; GPT = provider default; DeepSeek = provider default; Gemini = provider default). The average length per generation was 28.4 steps with a standard deviation of 12.4 steps. The average generation took 24.0 minutes with a standard deviation of 6.9 minutes.

# C Synthetic brief pipeline

## C.1 Brief generation prompt

**Meta-prompt for authoring briefs from websites (Claude Fable 5.1)**

```
System:
You are a design-brief author for a visual-generation study. You are shown one screenshot: the
    above-the-fold viewport of a real website's landing page. Distill it into a natural-language
    design brief that a web-building agent could implement without ever seeing the screenshot.

Anonymize completely: never name the company, product, brand, wordmark, domain, or any string
    visible in the screenshot that identifies the brand. Describe the business generically (for
    example "a financial-infrastructure platform"). Do not use the words Stripe, Linear, Figma,
    Shopify, or Airbnb anywhere, in any casing, even as ordinary adjectives.

Output only the brief text - no preamble, no headings, no markdown, no commentary.

User:
Write the design brief for the page in this screenshot as exactly three paragraphs separated by
    blank lines:

1. One sentence: "Design and build the landing page for <anonymized description of the business
    and its audience>, in a <two-or-three-adjective> register."
2. A paragraph beginning "Visual language:" covering the ground and ink colors, the accent
    color(s), the page's single most distinctive visual gesture, the typography's family
    character and weight, and the hero headline's treatment.
3. A paragraph beginning "Layout and compositions:" covering the navigation bar, the hero's
    contents and calls to action, what is visible below or beside the hero, and the spacing and
    alignment character.

Target length: about 500 words total. Name concrete counts, relative sizes, corner radii, shadow
    and gradient character, exact-ish color descriptions, and per-section content structure,
    while staying strictly anonymized.
```

[Image input: Screenshot of the web page is inserted here]

## C.2 Sample prompt

**Sample generated prompt from the Kumon website)**

Design and build the landing page for a supplementary children's education franchise offering after-school maths and English tuition programmes to parents of school-age children, in a warm, friendly, trustworthy register.

Visual language: The page sits on a clean white ground with near-black ink (a soft charcoal, roughly #1a1a2e) for headings and body text. The primary accent is a saturated marigold yellow-orange used for the main CTA banner band and header buttons; supporting accents include a bright periwinkle blue-purple (used for highlighted headline words and the primary button), a candy pink (used for a secondary header pill and a benefit icon disc), and a sky-blue used for a thin divider stripe under the header and a floating bottom promo bar. The single most distinctive visual gesture is the hero headline's two-tone treatment: a large serif-flavoured display sentence in charcoal with two emotive keywords ("skills" and "confidence" equivalents) rendered in the periwinkle accent, accompanied by a hand-drawn curved arrow doodle in the same purple sweeping up-and-right beneath the CTA button, pointing toward two programme cards. Typography is a rounded, humanist sans-serif with generous x-height; headings are bold (700-800 weight), body text regular, and the headline sits around 44-48px across two lines with a smaller 20px welcome line above it. Iconography is thin-stroke line art (books, calculators, pencils) in charcoal, giving a hand-crafted, approachable schoolroom feel.

Layout and compositions: The header is white and two-tiered: the top tier holds a playful logotype at left (with a smiley-face motif worked into one letterform), a centred rounded search module on a marigold pill background labelled with a "find your local instructor" prompt plus a circular search-icon button, then two rounded-rectangle CTA buttons at right -- one marigold ("Free Registration" with a book icon) and one pink ("Open a franchise" with a building icon), both roughly 12px corner radius. The second tier is a horizontal nav of five text links (two study-subject links, a news link, a helpdesk link, and an "About" dropdown with caret), underlined by a thin sky-blue rule. The hero is a full-width band (~460px tall) with a softly faded photograph of children studying bleeding in from the left at low opacity; left column holds the welcome eyebrow, the two-line two-tone headline, a pill-shaped periwinkle "Find out more" button (~195×48px), and the hand-drawn arrow doodle. To the right sit two identical white programme cards (~210×290px, ~24px radius, soft diffuse shadow), each containing a centred line-art icon (calculator-and-pencil for maths; book for English), a two-line label ("Maths study programme" / "English study programme"), and a circular charcoal arrow button (~48px) at the bottom. Directly below runs a full-width marigold banner (~100px tall) with painterly darker-orange brushstroke texture at its right edge, containing a line-art book icon, a bold ~28px heading inviting a free registration booking, and a dark charcoal pill "Book now" button. Beneath that begins a "Why choose us?" section: a bold two-line heading at left separated by a thin vertical rule from a row of three benefit items, each pairing a pastel circular icon disc (pink check, muted-yellow currency coin, lavender building) with a short two-line claim (multiple centre locations; low daily price from about £2.45; 600+ centres nationwide). A floating sky-blue rounded promo bar (~64px tall, large radius) hovers fixed near the bottom edge announcing a limited-time free-registration offer with a marigold outlined "Register for free" pill at right. Spacing is airy and grid-aligned to a centred ~1240px container, with generous 60-80px vertical rhythm between sections and consistent left alignment of textual content.

| Category | Eligible sites ($n = 58{,}220$) | web references ($n = 1{,}000$) | Brief sources ($n = 500$) |
|---|---|---|---|
| E-commerce & Retail | 34.7% | 35.0% | 36.2% |
| SaaS & Software Product | 12.8% | 14.7% | 15.2% |
| Education | 9.2% | 9.7% | 10.6% |
| Entertainment & Streaming | 5.6% | 7.5% | 8.4% |
| Corporate & Business | 6.6% | 6.3% | 7.0% |
| News & Media | 5.5% | 4.0% | 4.0% |
| Government & Public | 4.1% | 4.0% | 3.8% |
| Blog & Personal | 3.5% | 3.4% | 3.6% |
| Finance & Banking | 3.7% | 3.5% | 3.4% |
| Travel & Hospitality | 4.3% | 3.0% | 3.0% |
| Health & Medical | 3.1% | 3.0% | 2.4% |
| Reference & Documentation | 3.2% | 3.0% | 2.0% |
| Forum & Community | 2.1% | 1.5% | 0.4% |
| Other | 0.7% | 0.8% | 0.0% |
| Portal & Search | 0.8% | 0.6% | 0.0% |

Table 2: Distribution of semantic categorie of the briefs and generations.

## C.3 Distribution of semantic categories

## C.4 Design brief analysis methods

To analyze our briefs by the design dimensions they cover. We employ a two-stage process of summarization and then "clustering." We first employ Sonnet to give a summary of the core topics in a given brief, then we employ Haiku to go through these bulleted summaries to group them into distinct buckets. The goal is to understand the space of themes without succumbing to wording variations. This procedure is inspired by the Clio research project, aiming to model Claude conversations by topics [26].

**Theme-generation prompt (Claude Sonnet 5)**

```
These are clauses from website design briefs. Derive a canonical taxonomy of the THEMES these
    clauses specify: give each theme a short name and a one-sentence definition, keep themes
    mutually distinguishable, and cover about 95\% of the content. Ground every theme in what
    actually appears. Respond as JSON: {"themes": [{"name": "...", "definition": "..."}]}

[Stratified sample of 240 clauses, separated by ---]
```

**Clause-tagging prompt (Claude Haiku 5.5)**

```
Map each numbered clause to 1-3 themes from this fixed list (names only, best fit first; never
    invent a theme):
[Theme names and definitions from the first step]

Respond as JSON: {"mapping": {"<number>": ["theme name", ...]}}

[Numbered clauses, 40 per batch]
```

**Theme-grouping prompt (Claude Sonnet 5)**

```
Group these themes by the design dimension each belongs to, each theme under exactly one
    dimension. Give each dimension a short name and a one-sentence definition. Respond as JSON:
    {"dimensions": [{"name": "...", "definition": "...", "themes": ["theme name", ...]}]}

[Theme names and definitions from the first step]
```

| Dimension (share) | Model-generated definition |
|---|---|
| Visual Style (36.7%) | "Specifications of dominant background tones, ink/text colors, and accent hues (often with hex values) used for branding, highlights, and interactive elements."<br>"Descriptions of typeface style (serif, sans-serif, blackletter, etc.), weight, size, and case used for headlines, body copy, and labels." |
| Structural Layout (25.3%) | "Structural descriptions of how the page is divided into columns, containers, panels, or grids, including widths and alignment."<br>"Characterizations of whitespace, margins, gutters, and overall rhythm as airy, compact, generous, or tight."<br>"Descriptions of the composition and stacking of top header bars and bottom footer rows." |
| Content Modules (21.5%) | "Descriptions of the hero banner's headline, imagery, scrim, and overall arrangement at the top of the page."<br>"Descriptions of discrete card, tile, or panel compositions that group related content such as products, articles, or stats." |
| Navigation & Interaction (20.7%) | "Descriptions of navigation bars, link lists, menus, dropdowns, and their arrangement or styling."<br>"Styling and labeling details for primary and secondary action buttons, pills, and CTAs." |
| Decorative & Supporting Details (18.7%) | "Non-functional visual flourishes like gradients, overlays, watermarks, glows, or geometric patterns layered into the design."<br>"Thin lines, rules, or border treatments used to separate sections or frame elements."<br>"Small supporting UI elements such as icons, badges, bullets, dots, and toggles." |
| Imagery & Branding (11.3%) | "Descriptions of photographic subjects, illustrated artwork, or stock imagery used decoratively or informationally."<br>"Descriptions of the logo, wordmark, or icon mark representing the brand, including placement and style." |
| Strategic Context (11.2%) | "High-level statements defining the website's function, industry, and intended user base." |

Table 3: Extracted design dimensions from the model brief analysis pipeline. Design dimensions are ordered by the share of clauses belonging to them, from a total pool of 28,908 clause occurrences. Dimension share represents the percentage of clauses attributed to a given design dimension. Claude Sonnet 5 was used in the first stage of the pipeline and produced the theme definitions quoted in the second column. Claude Haiku 5.5 was used in the second stage to assign clauses to themes, and Claude Sonnet 5 was used in the third stage to group themes into dimensions.

# D Comparison of visual encoders

| Encoder pair | Debiased linear CKA | Shared top-10 neighbors (%) |
|---|---|---|
| SigLIP2–DINOv2 | 0.407 | 29.3 |
| SigLIP2–OpenCLIP | 0.727 | 52.7 |
| SigLIP2–UIClip | 0.604 | 42.5 |
| DINOv2–OpenCLIP | 0.357 | 28.7 |
| DINOv2–UIClip | 0.423 | 31.1 |
| OpenCLIP–UIClip | 0.547 | 41.6 |

Table 4: Agreement between visual encoders on page similarities where we look at the 100 page holdout for all developer groups. All LLM generations are from the set that had a matching web reference as the input for the brief.

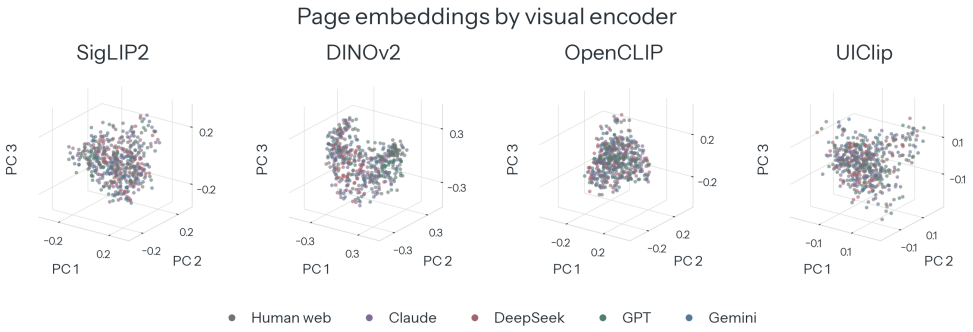


Figure 9: Comparison of visual encoders in captured 3D space.

| | Also found in (%) | | | | |
|---|---|---|---|---|---|
| Compositions from | Matched web | GPT | Claude | DeepSeek | Gemini |
| Matched web | 65 | 13 | 25 | 12 | 10 |
| GPT | 29 | 61 | 37 | 23 | 19 |
| Claude | 32 | 24 | 62 | 28 | 17 |
| DeepSeek | 35 | 32 | 57 | 62 | 34 |
| Gemini | 55 | 53 | 68 | 63 | 62 |

Table 5: Repeated compositions between developers.