January 2026

IAB EUROPE'S AI PROMPTING GUIDE

iabeurope.eu

iab europe

Introduction

The rise of artificial intelligence (AI) has fundamentally reshaped the digital advertising landscape, offering new ways to create content, target audiences, and optimise campaigns. At IAB Europe, we recognise that the effective use of AI is no longer a luxury but a necessity. In our July 2025 whitepaper, we highlighted that AI revenue is forecast to soar from roughly $200 billion in 2023 to about $1.4 trillion by 2029, signalling a vast and rapidly expanding commercial ecosystem.[1] And crucially for labour markets, 61 per cent of jobs are expected to be augmented - made more productive - by generative AI, with only 6.5 per cent likely to require long-term transition in Europe.[2]

Developed by IAB Europe's AI in Advertising Working Group, this guide is designed to empower digital advertising professionals by demystifying the art of **prompting**, a key skill for unlocking the full potential of AI models.

Why is prompting an important skill today?

Prompting is the art of communicating with an AI model to achieve a desired outcome. The quality of an AI's response is directly tied to the quality of the prompt it receives. A vague or poorly constructed prompt will likely produce a generic or irrelevant response, wasting time and effort. Conversely, a well-crafted prompt, one that is clear, specific, and provides sufficient context, can unlock creative and effective solutions. Knowing how to craft effective prompts allows digital advertising professionals to unlock better results from today's tools and future-proof their skills as AI becomes more deeply embedded in the workflows of the industry. In the fast-paced world of digital advertising, mastering prompting means you can:

- **Generate creative at scale:** Quickly produce multiple ad variations, subject lines, and social media posts.
- **Improve efficiency:** Automate routine tasks like keyword research or performance analysis.
- **Enhance strategy:** Use AI to brainstorm new campaign ideas and identify emerging trends.
- **Uncover insights:** Use AI to make data more discoverable and extract valuable insights for business development and performance optimisation.

[1] https://iabeurope.eu/wp-content/uploads/IAB-Europe-AI-Whitepaper-July-2025.pdf
[2] McKinsey Global Institute. "The Race to Deploy Generative AI and Raise Skills." McKinsey, 21 May 2024, www.mckinsey.com/mgi/our-research/a-new-future-of-work-the-race-to-deploy-ai-and-raiseskills-in-europe-and-beyond.

Who is this guidance for?

The tips and tricks in this guide are directed towards professionals across the digital advertising supply chain and, in most cases, are generally applicable across all kinds of generative AI use cases. They are categorised across **three levels of technical expertise** and colour-coded throughout the document. A description of the three levels is provided below.

Basic	Intermediate	Advanced
Structured use of conversational AI tools (e.g. prompt framing, iteration, role prompting, output evaluation) via chat-based interfaces.	Programmatic use of model APIs, prompt templating, workflow automation, and lightweight AI integrations within existing systems.	Hosting and fine-tuning open-source models, developing agentic systems, managing complex AI architectures, and addressing scalability, governance, security, and evaluation.

How can I use this guidance?

This guidance is designed to be a **practical reference** you can return to repeatedly as your use of AI tools evolves, applying different levels of guidance depending on the task, context, and constraints you are working within. At a foundational level, the guidance helps you become more **deliberate and effective** when interacting with AI systems. By applying the techniques and principles described, you can improve the quality, consistency, and usefulness of outputs across a wide range of workflows, including:

- generating and refining creative ideas,
- exploring and summarising data sets,
- supporting strategic thinking and scenario planning,
- accelerating research, documentation, and analysis,
- and enhancing day-to-day productivity in roles that rely on natural language interfaces.

As you move beyond individual interactions, the guidance supports more **repeatable and scalable usage**. Across all levels, the guidance also aims to build **shared literacy**. By using consistent terminology and concepts, it helps teams communicate more effectively about AI-related decisions, align expectations between technical and non-technical stakeholders, and assess trade-offs with greater clarity.

Definitions

Term	Definition
AI model	A computational system trained on data to recognise patterns and generate outputs, such as predictions, classifications, or text, based on given inputs.
Large language model (LLM)	A type of AI model trained on vast amounts of text data to understand and generate natural language by predicting the most likely sequence of words or tokens.
Token	A unit of text processed by a language model, typically representing a word, part of a word, or punctuation, used internally for computation and output generation.
Context window	The maximum amount of text (measured in tokens) that a model can consider at once when generating a response, including prompts, instructions, and prior conversation.
System prompt	A high-priority instruction that sets the overall behaviour, role, or constraints of a model throughout an interaction or session.
User prompt	The input or instruction provided by the user that directs the model to perform a specific task or generate a particular output.
Hallucination	A failure mode in which a model produces information that may be fluent and plausible but is factually incorrect, unsupported, or fabricated.
Inference	The process by which a trained model generates outputs from inputs, applying learned parameters without further training or modification.

Term	Definition
Vector	A numerical representation of data in a multi-dimensional space, used by models to encode meaning, relationships, or similarity.
Embedding	A specific type of vector that represents the semantic meaning of text, images, or other data, enabling comparison, clustering, and retrieval based on similarity.
Document store	A structured collection of documents or text chunks stored for retrieval, often used to supply external information to a model during generation.
Retrieval-augmented generation (RAG)	A system design pattern where a model retrieves relevant information from a document store and uses it as additional context when generating a response.
Tool calling	A capability that allows a model to invoke external functions, APIs, or services during inference in order to perform actions or retrieve structured data.
Agent	A system that combines a model with memory, tools, and decision logic to autonomously plan, execute, and adapt actions in pursuit of a goal.
Fine-tuning	A training process in which a pre-trained model is further trained on a specific dataset to adjust its behaviour or performance for particular tasks or domains.
Model drift	The gradual degradation or change in model performance over time due to shifts in data, context, usage patterns, or external dependencies.

Be Specific and Descriptive

To get the most out of prompting, **it's crucial to be intentional and structured in your requests**. If you're generating creative assets, think of it as providing a clear brief to an artist; the more detailed and specific the brief, the better the final result. Vague prompts like "make a cool ad" will yield generic results. Instead, specify the **subject, action, style, mood, and context**. For a video, you can include camera angles and lighting.

Bad Prompt	*Good Prompt*
`Make a video about a laptop.`	`A 10-second cinematic video showcasing a sleek, silver laptop. The camera slowly zooms in on the keyboard as a pair of hands gracefully types. The setting is a minimalist, sunlit home office with a plant in the background. The mood is calm and productive.`

Beyond describing the subject, you can explore options for other parameters related to photography to achieve the desired effect.

- **Camera Type / Lens** (DSLR, 35mm film, wide-angle, telephoto, tilt-shift) - Defines the "look" and technical character of the image.
- **Camera Angle / Perspective** (Eye-level, bird's-eye view, worm's-eye view, Dutch angle) - Alters how the subject is framed and perceived emotionally.
- **Focal Length / Depth of Field** (24mm wide, 85mm portrait, shallow depth of field, bokeh background) - Adjusts background blur and subject emphasis.
- **Lighting Style** (Natural daylight, studio softbox, golden hour, neon-lit, high contrast) - Shapes mood, realism, and dramatic effect.
- **Colour Grading / Palette** (Warm tones, cinematic teal-and-orange, pastel colours, monochrome) - Influences emotional tone and brand consistency.
- **Composition Style** (Rule of thirds, centred, minimalistic, busy/chaotic, negative space) - Controls visual balance and layout for storytelling.
- **Texture / Material Emphasis** (Glossy plastic, brushed metal, soft fabric, organic textures) - Helps communicate product qualities in ads.

Balance Positive and Negative Instructions

When giving instructions, it helps to balance what you want with what you want to avoid. Positive instructions set direction and intent; negative instructions act as guardrails. Problems arise when prompts rely too heavily on negatives, leading to generic or overly cautious output. The model spends effort avoiding mistakes instead of focusing on producing strong, relevant content. Negative instructions work best when they are specific, limited, and paired with clear positive guidance that anchors tone, purpose, and audience.

A good rule of thumb is to lead with positive instructions that describe the desired outcome, then add only the minimum number of negative constraints needed to prevent known issues. If you can phrase a requirement as a positive preference rather than a prohibition, it usually produces better results.

Bad System Prompt	*Good System Prompt*
`Write copy for a banner ad. Do not use hype. Do not sound salesy. Do not use buzzwords. Do not make promises. Do not be informal. Do not exaggerate. Do not mention discounts.`	`Write concise copy for a digital banner ad aimed at professionals. Use a clear, confident tone and focus on the main benefit in plain language. Keep it factual and easy to scan. Avoid exaggerated claims, marketing buzzwords, and overly promotional language.`

Provide a Role and Context

One of the simplest ways to improve results is to **tell the chatbot who it should be and what situation it is operating in**. This helps it choose the right tone, level of detail, and style automatically.

Many chatbot applications allow you to set a **system prompt** (sometimes called *instructions*, *custom instructions*, or *behaviour settings*). This is a persistent message that guides how the chatbot responds across the entire conversation. In the system prompt, you could specify things like:

- **Role or persona** (e.g. marketer, analyst, teacher, copywriter)
- **Domain or subject area** (e.g. digital advertising, finance, software)

- **Tone of voice** (e.g. professional, friendly, concise, persuasive)
- **Level of detail** (e.g. beginner-friendly, high-level, step-by-step)
- **Audience** (e.g. non-technical users, executives, students)
- **Style preferences** (e.g. bullet points, examples first, short paragraphs)

Bad System Prompt	*Good System Prompt*
`Be helpful and answer my questions.`	`You are a creative director working on digital advertising campaigns for premium consumer brands. Communicate clearly, use concise professional language, and prioritise practical, real-world examples.`

Multi-shot prompting

Multi-shot prompting involves giving the model several concrete examples of the task before asking it to perform the task itself. Rather than relying on a single instruction, you demonstrate the pattern you want the model to follow. This is particularly effective for structured extraction tasks, where the model needs to decide what information to include, how to label it, and what to do when information is missing.

By providing examples, you reduce ambiguity around interpretation and output format. This often leads to more consistent and reliable results, especially when extracting structured data from unstructured inputs such as ad copy, creative descriptions, or briefs.

Bad System Prompt

```
Extract the metadata from the ad below and return it as JSON.
```

Good System Prompt

```
I want you to extract metadata from advertising copy and return it as a JSON
object with the following fields:
 – brand
 – product
 – call_to_action

Here are two examples.

Example 1
 Input ad:
 ‘Discover Sparkle Soda. A refreshing lemon drink, now available online.
Order today.’
 Output JSON:
 {
 "brand": "Sparkle",
 "product": "Sparkle Soda (lemon)",
 "call_to_action": "Order today"
 }

Example 2
 Input ad:
 ‘Introducing NightRun trainers, designed for comfort and speed.’
 Output JSON:
 {
 "brand": null,
 "product": "NightRun trainers",
 "call_to_action": null
 }

Now apply the same logic to the ad below.
 Use exactly the same JSON fields.
 If a field is not present, return null.
 Do not add additional fields.

Ad copy:
 [PASTE AD COPY HERE]
```

Iterate and Refine

The first response is not always the final answer, especially when you are asking the model to analyse data, summarise information, or draw out insights. Treat the initial output as a starting point that helps you see what the model has understood and where it may need more direction.

If the overall approach is sound but something is missing or imprecise, you do not need to restate the entire request. Instead, build on the existing answer and be explicit about what should change. This keeps the context intact and usually leads to more accurate results.

Weak Follow-up	*Strong Follow-up*
`Can you improve this?`	`This summary is useful, but focus more on trends over time rather than individual data points, and call out any anomalies you see in the last quarter.`

Iteration is particularly effective when you refine one dimension at a time. You might ask the model to go deeper on a specific section, change the level of detail, apply a different analytical lens, or restructure the output for a specific audience. Each step clarifies your intent and reduces ambiguity.

The key idea is that prompting is an interactive process. Using successive, targeted refinements is often faster and more reliable than trying to encode every requirement perfectly in a single prompt.

Asking for Intermediate Prompts

In some cases, the tool you are using may rewrite or expand your request before passing it on to another system, such as an image generator, video generator, or other specialised capability. This means the instruction that actually gets used can be different from the one you originally typed.

When this happens, it can be useful to ask the model to **show you the exact prompt or instruction it plans to use before it proceeds**. Reviewing this intermediate step gives you a chance to check that your intent has been understood correctly and to

improve the wording directly, rather than guessing how your original prompt might be interpreted.

For example, you can ask the model to first present the prompt it intends to use, and only continue once you confirm it looks right. This allows you to adjust tone, level of detail, constraints, or emphasis before anything is generated or executed.

Example Prompt

```
You are an assistant that helps users generate content with
image or video generation tools. When the user gives you
instructions, you must first construct the exact prompt you
intend to send to the generation tool. Present this prompt back
to the user for confirmation in plain text, clearly marked as
Proposed Prompt. Do not proceed to generate or call any
external tool until the user explicitly confirms. Once
confirmed, you may then use the prompt with the generation
tool.
```

This approach is especially helpful when results feel close but not quite right. Instead of repeatedly tweaking your original prompt, you can refine the version that actually drives the output, which is often faster and more predictable.

Using Prompt Templates

Similar to system prompts, prompt templates are useful when you find yourself making similar requests repeatedly or needing consistent outputs across different tasks. Instead of rewriting prompts each time, you define a reusable structure and change only the parts that vary. This helps reduce ambiguity and improves consistency without requiring technical integration or automation.

A good way to think about a prompt template is as a fixed frame around a variable input. The frame captures things that should stay the same, such as the role the model should adopt, the scope of the task, and how the output should be structured. The variable part contains the specific data or question you want analysed.

Templates are particularly helpful when outputs are compared, reviewed by others, or used as part of an ongoing workflow. By keeping the structure stable, differences in output are more likely to reflect differences in input rather than differences in how the request was phrased. Templates should remain simple. If a template becomes difficult

to read or requires constant tweaking, it is usually a sign that too much logic has been pushed into the prompt instead of being handled through clearer inputs or follow-up questions.

Example Prompt Template

```
You are helping to name a conference panel. Based only on the
description below, propose 5 concise panel titles. The titles
should be clear, professional, and suitable for an industry
audience. Avoid marketing language and buzzwords.

Panel description: [PASTE PANEL DESCRIPTION HERE]
```

System Prompts as Agent Definitions

When building automated systems, the system prompt is often used to establish things like:

- **Identity & scope** (e.g. analyser, classifier, generator)
- **Instructions** (e.g. how to handle bad input, output constraints, verbosity)
- **Guardrails** (e.g. instructions not to hallucinate, flag uncertainty, content restrictions)

While system prompts can reduce risk, improve consistency, and shape how the model behaves quite effectively, alone they do not guarantee the desired outcome and should be paired with input / output validation and human review depending on the use case.

Bad System Prompt

```
You are an AI that checks ads.txt lines for errors. Check the ads.txt line
provided and return ‘true’ if malformed or ‘false’ otherwise.
```

Good System Prompt

```
You are a validation agent used in an automated ad tech workflow. Your task
is to evaluate one single line from an ads.txt file and determine whether it
is malformed according to the ads.txt specification.
```

You must return only a boolean value:
– true → the line is malformed
- false → the line is not malformed

Do not explain your reasoning. Do not return any additional text.

Rules you must apply
1. Pre-processing
- Ignore any content following a # character (inline comments are allowed).
- Evaluate only the portion of the line before #.

2. Line type
- Blank lines are not malformed.
- Comment-only lines are not malformed.
- All other lines are treated as data record lines.

3. Data record structure
- A data record must contain 3 or 4 comma-separated fields.
- Fields must be separated by commas (,), not other delimiters.

4. Field rules
- Field 1 (ad system domain): must be non-empty and contain no spaces.
- Field 2 (publisher account ID): must be non-empty.
- Field 3 (relationship type): must be exactly DIRECT or RESELLER (case-insensitive).
- Field 4 (optional): if present, must be non-empty.

5. Malformed if
- Fewer than 3 or more than 4 fields are present.
- Required fields are missing or empty.
- Invalid delimiters are used.
- The relationship type is not DIRECT or RESELLER.
- The line contains invalid or non-printable characters in the data portion.

6. Explicit non-errors
- Unknown domains or account IDs are not errors.
- Business correctness must not be evaluated.
- Do not infer intent or attempt correction.

Output contract (strict)
Return only:
- true
- false

No punctuation, no explanation, no formatting.

Most chat-style APIs support an explicit **system message** that persists across turns. Some APIs may simply prepend the system prompt to the user prompt without any structural distinctions.

Structure Prompts with JSON

Using JSON to structure prompts is helpful when you want stability, repeatability, and the ability to evolve prompts over time without losing control. Instead of treating a prompt as a single block of text, you treat it as structured data with clearly defined fields. This makes prompts easier to version, review, compare, and reuse across different tasks or users.

JSON prompts can be useful when a prompt needs to be refined incrementally, shared across a team, or passed through multiple steps in a workflow. Small changes can be made to individual fields without rewriting the entire prompt, and older versions can be kept alongside newer ones for reference or rollback. You can ask the model to return prompts in JSON that follow a defined schema, or to convert a standard JSON prompt schema into a text prompt before execution, which helps make prompt changes more deliberate and reduces unintended modifications to key constraints.

```JSON
{
  "version": "1.0",
  "intent": "image",
  "content": {
    "subject": "Soft drink can with condensation on glossy surface",
    "scene": {
      "environment": "Urban rooftop at sunset",
      "time_of_day": "golden_hour",
      "weather": "clear"
    },
    "composition": {
      "framing": "rule_of_thirds",
      "camera": {
        "type": "DSLR",
        "focal_length_mm": 50,
        "angle": "eye_level",
        "depth_of_field": "shallow"
      }
    },
    "lighting": {
      "style": "studio_key_fill_rim",
```

```
      "key_intensity": "high",
      "fill_intensity": "low",
      "color_grading": "warm_cinematic"
    },
    "style": {
      "medium": "photorealistic",
      "finish": "premium_cinematic",
      "brand_palette": ["#E50914", "#111111", "#FFD166"]
    },
    "product": {
      "name": "FizzUp Classic",
      "packaging": "330ml can",
      "logo_lockup_required": true
    },
    "text_overlays": [
      {
        "text": "Refresh Your Edge",
        "font": "Sans-Serif Bold",
        "placement": "upper_left",
        "size": "headline",
        "color": "#FFFFFF",
        "drop_shadow": true
      }
    ]
  },
  "constraints": {
    "brand_guidelines_strict": true,
    "disallowed_elements": ["other brand logos", "explicit content"]
  }
}
```

Automatic Prompt Refinement

Apart from static inputs, prompts can also be treated as artefacts that can be evaluated, normalised, and improved before being passed to a downstream model. This is particularly valuable in systems where prompts originate from non-expert users, upstream applications, or external services, and where consistency, safety, or performance matters.

A basic pattern can involve a dedicated agent sitting between the user input and the execution model. Its sole responsibility is to review the incoming prompt and decide whether it should be passed through unchanged, modified, or rejected. The refinement agent does not perform the task itself; it operates as a control layer that improves

clarity, enforces constraints, and reduces known failure modes before another model is invoked.

Architecturally, this introduces a clear separation of concerns. One component is responsible for intent normalisation and quality control, while another is responsible for task execution. This separation makes systems easier to reason about, easier to evaluate, and easier to evolve. Changes to prompt quality rules can be made without retraining or redeploying the execution model, and execution models can be swapped without changing upstream behaviour.

The refinement agent typically applies a fixed set of transformations. These often include:

- clarifying ambiguous instructions
- enforcing required structure
- injecting missing but mandatory constraints
- normalising terminology
- removing instructions that violate system-level policies

Importantly, refinement should be additive and corrective, not creative. Its goal is to preserve user intent while making that intent explicit and unambiguous.

From a governance perspective, this pattern provides a natural audit point. By storing both the original prompt and the refined version, teams can analyse how user intent is being transformed, identify systematic issues in upstream inputs, and detect drift in refinement behaviour over time.

Because the refinement task is narrow and repeatable, it is often well suited to fine-tuning, but there are limits to its effectiveness. Automatic prompt refinement is not a security boundary and should not be treated as one. It cannot guarantee correctness, prevent all misuse, or replace deterministic validation. For critical constraints, enforcement must still happen in code, schemas, or downstream checks. The refinement agent's role is to reduce variance and error, not to eliminate risk.

Positive / Negative Prompts

Some models or tools expose separate control channels for inclusion and exclusion. This is most common when calling models through APIs, running models locally, or using specialised generation tools where the prompt is not a single block of text but a structured set of inputs. Not all models support negative prompts as a first-class feature. They are most commonly available in image, video, and other generative

models derived from diffusion architectures, as well as in some locally hosted or open-source setups.

When supported, negative prompts act as a soft bias rather than a hard constraint, influencing what the model avoids during generation. Because support varies by model and API, advanced users should treat negative prompts as an optional optimisation layer: valuable when available, but never a substitute for clear positive guidance, validation, or downstream filtering.

Using the right model

At a basic level, using the right model is about recognising that not every task needs the most capable or most expensive option available. Models vary in quality, speed, and cost, and larger models are not automatically better for every use case. For many everyday tasks, a smaller or faster model may produce results that are perfectly adequate. When assessing cost, it is also important to understand how pricing is expressed: usage is often quoted per million tokens (MTok), but the relationship between tokens and words varies by model and provider, so headline prices are not always directly comparable. Choosing a model that is "good enough" for the task helps keep usage efficient and sustainable.

At an **intermediate level**, model selection should be an empirical exercise. Instead of assuming which model is best, teams should test multiple models against the same task and evaluate them across cost, quality, speed, and risk. This includes experimenting with different model sizes and configurations, keeping in mind that reducing model size and adjusting quantisation can have different effects on output quality and performance. A smaller, well-chosen model may outperform a larger one for a specific task, especially when outputs are constrained or repetitive. Regular testing is essential, as model performance and pricing change over time and can shift the balance of what "right-sized" means.

At an **advanced level**, systems can be designed so that models are interchangeable rather than hard-coded. This applies both to switching between locally hosted models and third-party APIs, and to swapping between different providers or different models from the same platform. Building this flexibility into the architecture enables continuous testing, benchmarking, and optimisation without disrupting workflows. Teams can route tasks dynamically based on cost, latency, risk profile, or availability, and can adopt new models or configurations as they emerge. In this setup, selecting the right model is no longer a one-time decision, but an ongoing process supported by measurement.

Checking your organisational policies

Before selecting or deploying a model, it is important to understand whether your organisation has existing policies around the use of generative AI. These may include guidance on how data can be shared with external providers, which models or platforms are approved for use, and whether certain tools are restricted to specific

types of work. In many cases, organisations also have established partnerships that have already been reviewed and accepted by Legal, Privacy, or Security teams. Aligning model choice with these policies early helps avoid rework, reduces risk, and ensures that AI is adopted in a way that is consistent with wider organisational responsibilities.

Batch prompting

Batch prompting refers to submitting a collection of inputs to a model in a single, non-interactive request, rather than sending each input as a separate, real-time prompt. The model processes each item independently using the same prompt structure, returning a corresponding set of outputs. This approach is most effective when the task is well-defined, does not require follow-up clarification, and can be applied consistently across many inputs.

This technique is particularly useful for large-scale, repeatable work where responsiveness is not critical. Typical use cases include generating descriptions for many products, producing multiple variations of ad copy, or applying the same transformation or extraction logic across a dataset. By reducing the number of individual calls and removing conversational overhead, batch prompting can significantly improve throughput, reduce operational cost, and make large-volume tasks easier to manage and monitor.

Energy efficiency

All of the recommendations in this guide can support more energy-efficient use of AI models, insofar as they help users achieve their intended outcomes while minimizing unnecessary resource consumption. Detailed data on the carbon footprint of individual AI interactions is not consistently available and may not always provide actionable insight at the user level. Nevertheless, it is reasonable to assume that emissions can be influenced in two ways:

1. **Defining the scope of the intended outcome:** users decide how much responsibility to give the model, from generating high-level ideas to producing fully polished outputs. Broader scope generally means more computation, while narrower, more targeted use reduces it.
2. **Improving process efficiency:** applying good prompting practices, such as clearer instructions, better structure, and fewer trial-and-error iterations, reduces the number of model calls needed to reach a satisfactory result.

For example, a digital advertising team tasked with developing campaign video scripts could take two different approaches. In one case, they might prompt the model for multiple fully written scripts, revising each through repeated calls. In a more energy-efficient scenario, they might instead prompt the model to generate concise bullet-point outlines that capture the narrative arc, then expand those manually. By narrowing the AI's role, they reduce the compute required, while still benefiting from the model's creative input.

Equally important is being mindful of the broader strategy for how AI is used within advertising workflows. Decisions made at the task level can have second-order effects: repeatedly relying on AI for outputs that could be handled more efficiently by templates, automation, or human expertise may not only increase emissions unnecessarily but also add hidden costs in quality assurance and oversight. Conversely, using AI strategically for the stages of work where it adds the most value can reduce inefficiencies across the campaign lifecycle. By considering not just immediate outputs but also the downstream implications of how AI is integrated, teams can balance creative gains with both environmental and operational responsibility.

Using Parameters & Tools

Beyond the text of a prompt itself, many generative models expose additional settings and tools that influence how they behave and what they produce. In consumer-facing chat interfaces these options may be limited or hidden, but they become more visible and powerful when using more advanced interfaces or APIs. The explanations in this section focus on practical impact rather than technical implementation, and are intended to help users understand when and why these controls matter.

System prompt / role

The system prompt, sometimes referred to as the role, is **an instruction given to the model before any user input**. It sets the overall frame for the conversation by defining how the model should behave, what perspective it should adopt, and what kind of outputs are appropriate. Rather than telling the model what to do in a single request, it shapes how the model responds across all subsequent prompts.

In practical terms, the system prompt is where you establish context, tone, and boundaries. For simple use cases, this may be as straightforward as asking the model to act as a specific professional or to communicate in a particular style. In more structured workflows, the system prompt is often used to define responsibilities, constraints, and failure behaviour, ensuring more consistent and predictable outputs over time.

Temperature

Temperature controls **how much randomness the model applies** when choosing the next token. Lower values bias the model toward more likely continuations, while higher values increase variation and introduce less predictable outputs.

A low temperature does not guarantee fully deterministic or identical results across runs. Even at the lowest setting, small variations can still occur due to the underlying generation process and other parameters. In practice, temperature should be understood as a way to reduce variability rather than eliminate it. For tasks that require consistency, it is best used alongside clear prompts and structural constraints, not as a standalone solution.

Number of outputs

Some chat interfaces and most APIs allow you to request **multiple alternative outputs** in a single call. This is conceptually different from increasing **temperature** or changing

parameters like **top-p**, and is often a cleaner way to explore variation while keeping parameters stable.

Response format / output schema

Many APIs allow you to specify or strongly encourage a **particular output format**, such as JSON, a fixed schema, or a constrained structure. This has a much bigger impact on reliability than most sampling parameters and is especially important when outputs are consumed by other systems. Even when not enforced, explicitly declaring an output format materially improves consistency.

Max tokens

The max tokens parameter usually sets a **hard limit on how long the model's response can be**, measured in tokens rather than characters or words. Once this limit is reached, the model stops generating text immediately.

It is important to understand that this parameter does not meaningfully guide the model toward an appropriate or well-structured length. In most cases, it simply truncates the response when the limit is hit, which can result in incomplete sentences or cut-off ideas. For controlling length in a more natural way, it is usually more effective to specify the desired length directly in the prompt, and use max tokens only as a safety cap rather than a stylistic control.

Stop sequences

Stop sequences define **one or more patterns that cause the model to immediately stop** generating output when they are encountered. They act as explicit boundaries, preventing the model from continuing beyond a predefined point.

In practice, stop sequences are most useful for enforcing structural limits rather than content quality. They help ensure outputs end cleanly, avoid unintended spillover into additional text, and maintain predictable formatting when responses are consumed by other systems or processes.

Context window size

The context window size defines **how much information the model can actively consider at any one time**, including the system prompt, user inputs, and earlier parts of the conversation. Once this limit is exceeded, older content falls out of scope and no longer influences the model's output.

The practical impact is that the model does not have a persistent memory of the entire interaction. In longer or more complex tasks, important instructions, constraints, or details introduced earlier may be partially or entirely ignored once they fall outside the context window. This can lead to inconsistencies, missing requirements, or shifts in tone or focus unless key information is restated or managed deliberately.

Top-p

Top-p, sometimes called nucleus sampling, **limits the model's next token choice to a subset of the most probable options** whose combined probability exceeds a specified threshold. Instead of considering all possible next tokens, the model samples only from this dynamically sized subset.

In practice, adjusting top-p changes how conservative or expansive the model's outputs feel. Lower values narrow the range of possible continuations and produce more predictable phrasing, while higher values allow a broader range of expression without fully randomising the output. Top-p is often used to fine-tune variation while maintaining overall coherence.

Top-k

Top-k **limits the model to choosing the next token from a fixed number of the most likely options**. Unlike top-p, the size of this set does not change based on probability mass; it always considers exactly k candidates.

Changing top-k affects how tightly the model adheres to its most likely predictions. Smaller values lead to more uniform and repetitive outputs, while larger values introduce more variation by widening the pool of possible continuations. Top-k offers a more rigid form of control than top-p, and the two are often used together to shape the balance between consistency and diversity.

Presence penalty

The presence penalty **reduces the likelihood that the model will revisit topics or ideas it has already introduced**. It encourages the model to move on to new concepts without strictly forbidding repetition.

In advertising ideation, this can help when generating multiple concepts or directions, as it reduces the chance of the model repeatedly returning to the same themes and encourages broader exploration.

Frequency penalty

The frequency penalty **reduces repetition at the word and phrase level** by discouraging the model from reusing the same tokens within a single output.

This can be useful in longer-form copy, where it helps prevent overuse of key terms and keeps the language feeling natural rather than repetitive or formulaic.

Logit bias

Logit bias allows **specific words or tokens to be made more or less likely** during generation by adjusting their probability before the model selects the next token. This does not force inclusion or exclusion, but it shifts the model's preferences toward or away from particular terms.

In practice, logit bias can be used to subtly steer language choices when consistency matters. For example, it can increase the likelihood of preferred terminology or standard phrasing being used across outputs, or reduce the chance of less suitable wording appearing, without having to hard-code those terms directly into the prompt.

Tools

Many generative AI systems can be extended with tools that allow models to access information, perform actions, or interact with other systems rather than relying on text generation alone. Some of these tools are exposed directly in consumer chat interfaces, others are available through APIs, and many can be custom-built when developing agents or integrated systems from scratch.

The table below outlines common categories of tools you may encounter across these environments. The exact names, capabilities, and implementation details vary by platform, but the underlying patterns are consistent. Understanding what these tools do and when to use them helps clarify how models move from answering questions to participating in structured workflows and decision-making processes.

Tool name	Description	Typical use case example
Retrieval / document lookup	Allows the model to access and reference external documents or a predefined knowledge base at runtime.	Answering questions based on internal policies, research documents, or campaign guidelines without embedding all content in the prompt.
Search	Enables the model to query up-to-date or external information sources during generation.	Pulling recent market developments or confirming current terminology before generating analysis or summaries.
Function calling	Allows the model to return structured arguments that trigger a predefined function or API call.	Converting natural-language requests into structured inputs for analytics, databases, or workflow automation.
Code execution	Lets the model run code in a controlled environment and return results.	Analysing datasets, generating charts, or validating calculations based on provided inputs.
File input / output	Enables the model to read from or write to files such as PDFs, spreadsheets, or text documents.	Reviewing reports, extracting key points from briefs, or generating formatted deliverables.

Tool name	Description	Typical use case example
Image generation	Allows the model to generate images based on text instructions.	Creating visual concepts, mock-ups, or illustrative assets from written descriptions.
Image understanding	Enables the model to analyse and describe images provided as input.	Reviewing creative assets, extracting text from visuals, or identifying elements in existing ads.
Tool routing / agent tools	Allows the model to decide when and how to use multiple tools as part of a single task.	Coordinating search, retrieval, and summarisation steps to produce a final report.

Managing Risks & Pitfalls

As generative AI is used across a wider range of tasks, the nature of risk shifts with the way the technology is applied. At one end of the spectrum, managing risk may simply mean recognising low-quality or misleading output in a chat interface and choosing not to rely on it. At the other end, it involves designing systems that can safely incorporate model outputs into automated or decision-support workflows.

The risks discussed in this section apply across that spectrum. Some can be addressed through careful prompting, review, and user judgement, while others require structural safeguards such as validation, monitoring, and clear boundaries around how models are used. Understanding these risks helps users make better decisions about when to trust outputs, when to intervene, and when additional controls are necessary to ensure AI is used responsibly and reliably.

Hallucination

Definition

Hallucination occurs when a model generates information that is incorrect, misleading, or entirely fabricated, while presenting it with confidence. This can include invented facts, sources, relationships, or explanations that are not grounded in the provided input or any reliable external reference.

How to identify

Hallucinations often appear as specific details that cannot be verified, such as precise statistics without sources, citations that do not exist, or confident answers to questions that lack sufficient context. A common signal is when outputs go beyond the information supplied, especially in analytical or factual tasks. Inconsistent answers to the same question across multiple runs can also indicate hallucination risk.

Mitigation strategies

Limit hallucination by narrowing task scope and explicitly instructing the model not to guess or infer missing information. Require the model to flag uncertainty or return “unknown” when inputs are insufficient. For high-impact use cases, pair model outputs with human review, deterministic checks, or external data validation rather than relying on the model as a sole source of truth.

Context exhaustion

Definition

Context exhaustion occurs when the total amount of input and conversation history exceeds the model's context window. When this happens, earlier instructions, constraints, or information fall out of scope and no longer influence the model's output.

How to identify

Signs of context exhaustion include the model ignoring earlier requirements, changing tone or assumptions mid-task, repeating questions that were already answered, or producing outputs that contradict previous instructions. This is most common in long conversations, complex multi-step tasks, or workflows that accumulate large inputs over time.

Mitigation strategies

Keep prompts focused and avoid carrying unnecessary history forward. Restate critical constraints and objectives explicitly when tasks become long or complex. Where possible, summarise earlier content before continuing, or break large tasks into smaller, self-contained steps. In system design, manage context deliberately rather than assuming the model retains full conversational memory.

Bias

Definition

Bias arises when a model produces outputs that reflect unfair, stereotypical, or unbalanced assumptions about individuals or groups. This can be triggered by biased training data, poorly framed prompts, or unexamined defaults in how tasks are defined.

How to identify

Bias may appear as uneven treatment of groups, stereotypical language, exclusion of certain perspectives, or assumptions presented as facts. It can also surface indirectly, for example through consistently skewed recommendations or framing that favours one group or outcome without justification.

Mitigation strategies

Use neutral, precise language and avoid prompts that rely on implicit assumptions. Review outputs for patterns rather than isolated cases, especially when models are used at scale. Where fairness matters, test prompts across a range of inputs and scenarios, and introduce human oversight for sensitive or high-impact decisions.

Sensitive data exposure

Definition

Sensitive data exposure occurs when confidential, personal, or proprietary information is included in prompts or generated in outputs inappropriately. This can involve personal data, commercial secrets, or information subject to legal or contractual restrictions.

How to identify

Risk is present whenever prompts include identifiable personal details, internal documents, customer data, or non-public business information. Exposure may also occur if outputs echo or transform sensitive inputs in ways that make them reusable or shareable beyond their intended context.

Mitigation strategies

Remove or anonymise sensitive information before including it in prompts. Limit the amount of context shared to what is strictly necessary for the task. Follow organisational guidance on approved tools and data handling, and apply access controls where models are integrated into workflows. For sensitive use cases, ensure outputs are reviewed before being stored, shared, or acted upon.

Contributors

Thank you to the following members of the IAB Europe Artificial Intelligence in Advertising Working Group for their contributions to this guide:

- Google
- Microsoft Advertising
- Pubmatic
- MarkApp
- ShowHeroes
- BVDW
- IAB Finland
- SPIR
- IAB Austria

Dimitris Beis
Data Analyst & Sustainability Lead
beis@iabeurope.eu

iab europe
Rond-Point Robert
Schumanplein 11
1040 Brussels
Belgium

@iabeurope

/iab-europe

iabeurope.eu