PHILIPPE COHEN
AutomatisIA
Avril 2025

COMPRENDRE ET PROGRESSER

MAÎTRISER LE PROMPT ENGINEERING AVEC LES MODÈLES OPENAI

AUTOMATISIA
Optimiser les competences humaines

Introduction **2**

1 . Qu'est-ce que le prompt engineering ? — définition, importance, impacts **3**

1.1 Définition 3

1.2 Importance opérationnelle 3

1.3 Impacts sur le cycle de développement et les métiers 3

1.4 Quand l'optimisation est superflue 4

2 . Qualité des réponses & phénomène d'hallucination **5**

2.1 Définir une réponse « de qualité » 5

2.2 Hallucination : nature et manifestations 5

2.3 Facteurs qui dégradent la qualité 6

2.4 Méthodes de contrôle et de mitigation 6

2.5 Impact productivité : latence et arbitrage 7

3 . Techniques fondamentales de prompt-engineering **8**

3.1 Chain of Thought (CoT) 8

3.2 Self-Consistency 8

3.3 Injection de documents externes 8

3.4 ReAct (Reasoning + Acting) 9

3.5 Prompting inversé 9

3.6 Décomposition structurée des tâches 10

3.7 Formats structurés (XML, JSON, Markdown) 10

4 . Lexique des verbes, modificateurs et contextuels essentiels **11**

4.1 Verbes d'action — définir sans ambiguïté la tâche attendue 11

4.2 Modificateurs de qualité — affiner le rendu attendu 12

4.3 Marqueurs contextuels — encadrer la génération 12

4.4 Assembler les composants : démarche recommandée 13

5 . Choisir le bon modèle OpenAI et adapter son prompt **14**

5.1 Comparatif express des modèles disponibles 14

5.2 Fiches-modèles détaillées 15

5.2.1 GPT-4o 15

5.2.2 o3 15

5.2.3 o4-mini 16

5.2.4 o4-mini-high 16

5.2.5 GPT-4.5 17

5.2.6 GPT-4.1 18

6 . Conclusion & prochaines étapes **19**

6.1 Principaux enseignements 20

6.2 Bonnes pratiques transverses 21

6.3 Ressources complémentaires 22

Prochaines étapes 23

INTRODUCTION

Large Language Models (LLMs) tels que ceux d'OpenAI ont profondément modifié la production de texte, l'analyse de données et l'automatisation de tâches intellectuelles. Leur qualité dépend cependant presque entièrement de la manière dont l'utilisateur formule sa requête : c'est le **prompt engineering**. Bien qu'il s'agisse d'un concept récent, maîtriser cette discipline est devenu indispensable pour trois raisons :

1. Fiabilité – La formulation du prompt influence la précision factuelle, la structure et la cohérence du résultat.

2. Efficacité – Un prompt bien conçu réduit les cycles d'allers-retours, diminue la latence de génération et donc le coût opérationnel.

3. Traçabilité – Un schéma de prompt reproductible permet de contrôler la variabilité du modèle, de documenter les choix et de répondre aux exigences de conformité (auditabilité, explicabilité).

Le dossier qui suit propose :

- une définition précise du prompt engineering, ses impacts et ses limites ;
- un chapitre dédié à la qualité de réponse et aux hallucinations, ainsi qu'aux leviers pour les contenir ;
- un exposé détaillé des techniques fondamentales — de la chaîne de pensée à l'injection de documents externes ;
- un lexique opérationnel des verbes d'action, modificateurs de qualité et marqueurs contextuels les plus utiles ;
- enfin, un comparatif des principaux modèles OpenAI (GPT-4o, o3, o4-mini, o4-mini-high, GPT-4.5, GPT-4.1) assorti de recommandations de prompt spécifiques.

L'objectif est de fournir aux équipes produit, data et innovation un cadre systématique pour concevoir, tester et améliorer leurs prompts, qu'il s'agisse de tâches simples (traduction, correction) ou de workflows complexes (recherche documentaire, génération de code, analyses multimodales).

1 . QU'EST-CE QUE LE PROMPT ENGINEERING ? – DÉFINITION, IMPORTANCE, IMPACTS

1.1 DÉFINITION

Le *prompt engineering* est la conception méthodique des instructions adressées à un modèle de langage pour générer un résultat précis, fiable et reproductible. Il s'agit de choisir :

- **le rôle** que le modèle doit endosser (expert, assistant, auditeur, etc.) ;
- **la tâche** à exécuter, formulée par un verbe d'action clair ;
- **le contexte** pertinent (extraits de documents, contraintes métier, exemples) ; • **le format** de sortie attendu (prose, tableau, JSON, diff de code, etc.).

L'enjeu n'est pas uniquement linguistique : un prompt bien spécifié agit comme une interface de programmation sans code, guidant l'algorithme vers la solution attendue.

1.2 IMPORTANCE OPÉRATIONNELLE

1. **Précision et cohérence** : un prompt structuré réduit les erreurs factuelles et limite la variance entre deux exécutions.

2. **Productivité** : moins d'allers-retours signifie un cycle de travail plus court et un coût de calcul plus faible.

3. **Conformité** : expliciter les contraintes (RGPD, confidentialité, ton de marque) dans le prompt permet de documenter la décision et de répondre aux audits internes.

4. **Interopérabilité** : un prompt réutilisable garantit un comportement stable lorsque l'on change de modèle ou que l'on met à jour l'API.

1.3 IMPACTS SUR LE CYCLE DE DÉVELOPPEMENT ET LES MÉTIERS

- **Développeurs** : le prompt devient un artefact versionné au même titre que le code ; il doit être testé, validé et maintenu.
- **Data & IA** : le prompt complète (ou parfois remplace) la phase de fine-tuning pour ajuster le modèle à un cas d'usage.
- **Équipes métier** : la compétence "rédaction de prompt" se traduit par une meilleure autonomie pour prototyper des rapports, des analyses ou des supports marketing.
- **Gouvernance** : des conventions de prompt uniformes facilitent l'audit, la sécurité et la traçabilité des réponses produites par les LLM.

1.4 QUAND L'OPTIMISATION EST SUPERFLUE

Tous les usages ne justifient pas un prompt sophistiqué. Les cas suivants fonctionnent avec une formulation minimale :

- **Traduction directe** : « Traduis ce texte en anglais ».
- **Correction orthographique / grammaticale** : « Corrige ce paragraphe ».
- **Détection simple d'intention** : « Classifie ces e-mails (spam / non-spam) ».

Dans ces scénarios, ajouter des exemples ou imposer un format complexe n'améliore pas la qualité de manière significative. L'effort d'ingénierie serait donc une dépense de temps sans retour sur investissement.

En revanche, dès que la tâche exige :

- plusieurs contraintes simultanées (ton, longueur, structure) ;
- un raisonnement multi-étapes (analyse juridique, audit financier) ;
- l'intégration de sources externes ou d'outils (recherche web, exécution de code) ;

le prompt engineering devient un levier essentiel pour garantir la pertinence, la conformité et la reproductibilité des résultats.

2 . QUALITÉ DES RÉPONSES & PHÉNOMÈNE D'HALLUCINATION

2.1 DÉFINIR UNE RÉPONSE « DE QUALITÉ »

Une sortie produite par un LLM est considérée de bonne qualité lorsqu'elle réunit cinq critères simultanés :

- **Exactitude factuelle** : les données chiffrées, références et citations sont correctes.
- **Cohérence logique** : l'argumentation ne comporte pas de contradictions internes ni d'erreurs de raisonnement.
- **Complétude** : tous les points explicitement demandés dans le prompt sont couverts. • **Pertinence** : le contenu reste strictement aligné sur la question, sans digressions inutiles.
- **Lisibilité et format** : la structure répond aux contraintes (longueur, table, JSON, Markdown, etc.) fixées par l'utilisateur ou l'API.

Ces dimensions s'évaluent en priorité par validation humaine ou par des tests automatisés (regex, schémas JSON, suites d'evals).

2.2 HALLUCINATION : NATURE ET MANIFESTATIONS

On parle d'*hallucination* lorsqu'un modèle génère des informations plausibles mais factuellement incorrectes ou non fondées dans ses données d'entraînement. Trois formes majeures sont observées :

1. **Hallucination factuelle** : inventer des dates, chiffres, citations, URL, intitulés de lois.
2. **Hallucination structurelle** : fournir un format demandé mais dont les champs contiennent des valeurs artificielles (par exemple un tableau avec des références bibliographiques inexistantes).
3. **Hallucination de provenance** : affirmer que la réponse s'appuie sur une source précise (« Rapport ONU 2024 ») alors qu'aucune trace vérifiable n'existe.

2.3 FACTEURS QUI DÉGRADENT LA QUALITÉ

Plusieurs variables augmentent la probabilité d'erreurs ou d'hallucinations :

- **Prompt ambigu ou contradictoire** : consignes qui s'annulent (« sois concis » et, plus loin, « détaille chaque point »).
- **Contexte insuffisant ou bruité** : absence de données externes, liens morts, citations tronquées.
- **Dépassement de la capacité de raisonnement du modèle sollicité** : demander une preuve mathématique longue à un modèle non raisonneur.
- **Questions hors couverture temporelle** : interroger sur des faits postérieurs au *knowledge cut-off* sans fournir de mise à jour.
- **Contraintes de format trop lâches** : laisser au modèle le soin de deviner la structure finale induit des divergences entre exécutions.

2.4 MÉTHODES DE CONTRÔLE ET DE MITIGATION

Fournir la matière première : intégrer directement dans le prompt des extraits vérifiés ou recourir à l'injection de documents externes (vector store, file search) pour ancrer la génération dans une base de vérité.

Montrer plutôt que décrire : un ou deux exemples *few-shot* correctement formatés réduisent la variabilité et guident le modèle vers la réponse attendue.

Segmenter le raisonnement :

• pour les modèles raisonneurs (o3, o4-mini-high), demander explicitement une chaîne d'étapes (« Expose ta démarche avant la conclusion ») ;

• pour les modèles non raisonneurs (GPT-4 o, GPT-4.5), limiter ce type d'instructions afin de ne pas alourdir inutilement la sortie.

Imposer des validations automatiques :

• schémas JSON, expressions régulières, contrôles de longueur ;

• exécution de scripts externes (Python) pour vérifier une formule, recalculer un total, ou tester une réponse sur un petit jeu de données.

Itérer ou voter : lancer plusieurs générations indépendantes et comparer les convergences (self consistency) ; rejeter la réponse isolée qui diverge sans justification.

2.5 IMPACT PRODUCTIVITÉ : LATENCE ET ARBITRAGE

La qualité s'améliore souvent au prix d'un délai supplémentaire :

- Un appel à un modèle raisonneur (o-series) qui explicite son cheminement peut durer plusieurs dizaines de secondes.
- Les boucles de re-vérification (double génération, appels API externes, contrôle par script) allongent encore le temps de réponse.

Il convient donc de calibrer la profondeur du prompt selon l'usage :

- **requêtes interactives** (chat en temps réel, service client) → privilégier la réponse immédiate d'un GPT-4 o structuré, même s'il est moins exhaustif ;
- **analyses à forte valeur ajoutée** (audit, note de synthèse, rapport réglementaire) → accepter la latence induite par une procédure de vérification et, si nécessaire, planifier ces traitements hors session interactive.

En pratique, un compromis clair entre délai acceptable et niveau de sûreté doit être défini dans le cahier des charges ; il guidera le choix du modèle, la présence d'étapes de contrôle et le degré de détail exigé dans le prompt.

3 . TECHNIQUES FONDAMENTALES DE PROMPT-ENGINEERING

3.1 CHAIN OF THOUGHT (COT)

Principe : demander explicitement au modèle d'énoncer les étapes de raisonnement avant de livrer la conclusion.

Mise en œuvre : ajoutez, juste avant la question, une instruction courte : « Expose ton raisonnement étape par étape, puis conclus en deux phrases. »

Bénéfice : sur les modèles raisonneurs (o-series) la qualité logique et la transparence augmentent ; sur les modèles non raisonneurs (GPT-4 o, GPT-4.5) la mécanique reste utile, mais rallonge la réponse.

Limites : le texte justificatif coûte en temps de génération ; il faut l'écarter des livrables destinés à l'utilisateur final ou le tronquer après validation.

3.2 SELF-CONSISTENCY

Principe : exécuter le même prompt plusieurs fois, comparer les raisonnements internes et retenir la réponse la plus fréquente ou la plus argumentée.

Mise en œuvre : lancez N (3 à 5) générations en parallèle ; sélectionnez la sortie convergente ou combinez-les pour construire un consensus.

Bénéfice : réduit les erreurs arithmétiques et les contradictions sur des problèmes complexes.

Limites : multiplie d'autant la latence ; ne s'applique qu'aux tâches où plusieurs réponses possibles ne créent pas de confusion côté utilisateur.

3.3 INJECTION DE DOCUMENTS EXTERNES

Principe : fournir au modèle des segments de texte pertinents (moteur de recherche interne, vector store, file search) et l'instruire de s'y référer exclusivement.

Mise en œuvre :

1. Sélection des passages les plus proches de la requête.
2. Inclusion directe dans le prompt sous un délimiteur clair (« *### Contexte* »).

3. Directive : « Réponds uniquement à partir du contexte fourni ; cite la section lorsque tu relies une information. »

Bénéfice : abaisse drastiquement le taux d'hallucination et permet de travailler sur des données privées.

Limites : la pertinence dépend de l'algorithme de recherche ; un contexte trop volumineux dilue l'attention si le modèle n'est pas dimensionné pour le traiter.

3.4 REACT (REASONING + ACTING)

Principe : alterner raisonnement textuel et appel d'outils (code, recherche web, accès base de données). Le modèle planifie, agit, puis réfléchit à nouveau.

Mise en œuvre : adapter le prompt système :

```
Lorsque c'est nécessaire, appelle la fonction python.execute pour
vérifier un calcul ou web.search pour confirmer un fait, puis poursuis
le raisonnement.
```

Bénéfice : combine la créativité du LLM et la vérification externe ; idéal pour analyses chiffrées ou synthèses documentaires.

Limites : nécessite des droits d'appel d'outils et une orchestration fiable côté backend ; chaque cycle ajoute de la latence.

3.5 PROMPTING INVERSÉ

Principe : laisser le modèle inférer la question, puis reformuler soi-même la demande définitive. **Mise en œuvre** :

1. Envoyer un extrait de texte complexe.
2. Demander : « Quelle requête précise devrais-je te poser pour résumer correctement ce document ? »
3. Valider ou affiner la requête retournée.

Bénéfice : utile quand l'utilisateur ne sait pas formuler son besoin ; clarifie les attentes avant génération.

Limites : ajoute une étape de dialogue ; dépend de la capacité du modèle à détecter l'intention réelle.

3.6 DÉCOMPOSITION STRUCTURÉE DES TÂCHES

Principe : scinder une requête ambitieuse en sous-prompts sériels (plan → recherche → rédaction → relecture).

Mise en œuvre : pipeline de prompts, chaque étape recevant en entrée la sortie validée de la précédente.

Bénéfice : maîtrise fine des livrables intermédiaires, détection précoce d'erreurs.
Limites : pipeline plus long à exécuter ; requiert une supervision ou une automatisation pour chaîner les étapes.

3.7 FORMATS STRUCTURÉS (XML, JSON, MARKDOWN)

Principe : imposer un balisage clair pour que la sortie soit immédiatement exploitable par une machine ou un autre service.

Mise en œuvre :

```
Retourne la réponse au format JSON strict suivant : { "titre": "",
"arguments": [ { "id": 1, "texte": "" } ] }  Aucune autre sortie.
```

Bénéfice : supprime l'ambiguïté et facilite l'intégration dans un workflow d'API ou une base de données.

Limites : certains modèles bavards ajoutent du texte hors champ ; prévoir un validateur JSON et, au besoin, rappeler la contrainte dans un second tour.

4 . LEXIQUE DES VERBES, MODIFICATEURS ET CONTEXTUELS ESSENTIELS

Cette section rassemble les expressions les plus utiles pour structurer un prompt en français. Elles sont classées en trois familles : **verbes d'action** (précisent la tâche), **modificateurs de qualité** (affinent le rendu) et **marqueurs contextuels** (délimitent les contraintes). Chaque entrée est accompagnée d'un commentaire d'usage et d'un court exemple intégré dans une phrase-type ; l'objectif est de fournir un vocabulaire opérationnel immédiatement réutilisable.

4.1 VERBES D'ACTION – DÉFINIR SANS AMBIGUÏTÉ LA TÂCHE ATTENDUE

- **Analyser** – demander un examen critique ou un découpage logique. Exemple : « Analyse les écarts entre le budget prévisionnel et le réalisé. »
- **Comparer** – exiger une confrontation d'éléments avec critères implicites ou explicites. *« Compare les stratégies de prospection de ces trois concurrents. »*
- **Déduire** – contraindre le modèle à extraire une conclusion à partir d'un corpus. *« À partir de ce procès-verbal, déduis les risques légaux majeurs. »*
- **Résumer** – condenser l'information en préservant les points clés.
- « Résume en 150 mots ce chapitre de rapport technique. »
- **Reformuler** – changer le style ou le niveau sans altérer le fond.
- « Reformule ce mail dans un registre formel. »
- **Structurer** – imposer une organisation (sections, puces, JSON).
- « Structure ces notes de réunion en trois rubriques : décisions, actions, questions ouvertes. »
- **Vérifier** – solliciter une validation ou un contrôle qualité.
- « Vérifie la cohérence logique de cette argumentation. »
- **Proposer** – générer des pistes ou recommandations.
- « Propose deux scénarios de pricing pour un SaaS en phase de lancement. »

4.2 MODIFICATEURS DE QUALITÉ – AFFINER LE RENDU ATTENDU

• **Clarté** – invite à simplifier la syntaxe et à éviter le jargon.

« Explique clairement le principe du hashing à un public non technique. »

• **Précision** – encourage la concision des chiffres, dates, références.

« Détaille précisément les versions des normes ISO citées. »

• **Exhaustivité** – demande de couvrir toutes les facettes d'un sujet.

« Présente de manière exhaustive les étapes de la chaîne d'approvisionnement. »

• **Synthétique** – réduit la longueur sans sacrifier l'information critique. *« Fais un compte-rendu synthétique de 10 lignes maximum. »*• **Critique** – insiste sur l'évaluation (forces, faiblesses, biais).

« Évalue de façon critique la méthodologie de cette étude clinique. »

• **Didactique** – oriente la réponse vers un ton pédagogique, progressif. *« Explique de façon didactique l'algorithme de Dijkstra. »*

4.3 MARQUEURS CONTEXTUELS – ENCADRER LA GÉNÉRATION

• **En s'appuyant sur …** – force l'utilisation d'un support précis.

« En t'appuyant sur les articles 23-25 du RGPD, explique la base légale invoquée. »

• **Sans inventer de données** – rappelle la nécessité d'éviter l'hallucination. « Rédige la synthèse sans inventer de sources supplémentaires. »

• **En tenant compte des contraintes …** – introduit un cadre métier ou réglementaire. « En tenant compte des contraintes d'accessibilité WCAG 2.2, propose trois améliorations UI. »

• **Dans la limite de …** – fixe un seuil de longueur ou de détail.

« Dans la limite de 800 caractères, explique l'intérêt du chiffrement homomorphe. »

• **Pour une PME / pour un public X** – spécifie l'audience et ajuste le niveau de technicité. « Pour une PME industrielle, détaille les pré-requis d'un projet RPA. »

• **Format attendu : …** – bloque la structure finale.

« Format attendu : tableau Markdown à deux colonnes (risque, action). »

4.4 ASSEMBLER LES COMPOSANTS : DÉMARCHE RECOMMANDÉE

1. **Démarrer par le verbe d'action** afin de lever toute ambiguïté sur la tâche.
2. **Ajouter un modificateur de qualité** si la façon de répondre (détail, ton, longueur) est critique.
3. **Clore avec un marqueur contextuel** pour borner le périmètre, citer la source ou imposer le format.

Exemple complet :

« **Analyse** les écarts entre budget initial et dépenses réelles Q1-Q2 2024 **de façon critique**, **en t'appuyant sur** le fichier CSV fourni et **dans la limite de 15 lignes**. »

Cette construction hiérarchique (action → qualité → contexte) maximise la clarté pour le modèle et facilite la réutilisation du prompt dans des scénarios similaires.

5 . CHOISIR LE BON MODÈLE OPENAI ET ADAPTER SON PROMPT

5.1 COMPARATIF EXPRESS DES MODÈLES DISPONIBLES

Modèle	Contexte	Orientation principale	Temps de réponse moyen*	Cas d'usage typiques
GPT-4o	128 K	Polyvalence, robustesse ; suit bien les instructions explicites	≈ 5–7 s	Production générale de texte, chat interactif, traduction, support client
o3	200 K	Raisonnement séquentiel poussé, appels d'outils autonomes	≈ 15 s	Débogage, analyse scientifique, dossiers réglementaires complexes
o4-mini	200 K	Raisonnement à coût réduit, réponses concises	≈ 3 s	Q&A volumineux, automatisation back-office, prototypes rapides
o4-mini high	200 K	Même base qu'o4-mini mais plus de passes de réflexion	≈ 6 s	Codage intermédiaire, tâches visuelles légères, synthèses étoffées
GPT-4.5	128 K	Intuition linguistique, ton « humain », créativité narrative	≈ 8 s	Rédaction marketing premium, storytelling, assistance éditoriale
GPT-4.1	1 M	Suivi littéral d'instructions, génération de diff code	≈ 6 s	Agents développeurs, revues de code, traitements long-contexte

* Temps observés via ChatGPT Plus/Enterprise au 1er trimestre 2025, en entrée texte seule.

5.2 FICHES-MODÈLES DÉTAILLÉES

5.2.1 GPT-4o

Positionnement – modèle « par défaut » : rapide, équilibré, fiable sur les tâches généralistes.

Capacités clés – bonne compréhension contextuelle, multimodal sur demande, faible hallucination si prompt clair.

Pièges – moins nuancé qu'un GPT-4.5 sur l'émotionnel ; peut imposer des listes non sollicitées.

Outils – recherche web et exécution Python activables manuellement.

Structure de prompt recommandée :

> Rôle (optionnel) → tâche → contraintes de format → longueur souhaitée.
>
> **Exemple « prospection B2B sous tension »**
>
> « Rédige une note de cadrage (≈ 300 mots) exposant trois leviers de prospection B2B dans un secteur saturé. Mets en avant pour chaque levier : canal utilisé, KPI principal, risque. Sortie : tableau Markdown. »

5.2.2 o3

Positionnement – moteur de raisonnement premium.

Capacités – décomposition logique étape par étape, vérification via Python, lecture d'images complexes.

Pièges – verbeux par défaut ; latence élevée ; quotas plus stricts.

Outils – tous activés de façon agentique (web, code, vision).

Prompt type

> Contexte minimal → 2) « Analyse étape par étape » → 3) Format final court.
>
> **Exemple prospection**
>
> « Tu es consultant senior. Analyse étape par étape les freins actuels de la prospection B2B dans la cybersécurité, propose ensuite deux tactiques neuves validées par des cas réels (cite les sources web que tu trouves). Conclus en dix lignes max. »

Astuce productivité – Préciser *« 10 lignes max »* limite la verbosité et réduit la latence.

5.2.3 o4-mini

Positionnement – alternative économique quand le raisonnement reste souhaité.
Capacités – chaîne de pensée interne, vitesse supérieure à o3, coût modeste.

Pièges – moins apte à comprendre les implicites ; hallucination légèrement plus fréquente qu'o3.

Outils – identiques à o3 mais déclenchés moins souvent.

Prompt type

> Formulations courtes, directives explicites, format imposé.
>
> **Exemple prospection**
>
> « Liste trois canaux hors-ligne encore efficaces pour prospecter dans l'industrie pharmaceutique ; pour chaque canal, donne un ratio coût/lead (fourchette chiffrée) et un exemple de mise en œuvre. Format : liste numérotée. »

Astuce spécifique – Ajoutez « En cas d'incertitude, réponds : "données insuffisantes". » pour réduire l'invention de chiffres.

5.2.4 o4-mini-high

Positionnement – même architecture qu'o4-mini mais configurée pour réfléchir plus longtemps.

Capacités – meilleure pertinence sur questions analytiques, lecture d'images simple, génération de code court.

Pièges – latence doublée par rapport à o4-mini ; quotas plus serrés.

Outils – identiques à o4-mini.

Prompt type

> Préfixe : « Prends le temps d'analyser ; explique brièvement ta démarche, puis synthétise. »
>
> **Exemple prospection**
>
> « Prends le temps d'analyser la maturité LinkedIn Ads dans les PME industrielles françaises. Explique brièvement ta démarche (5 lignes), puis propose deux scénarios de prospection paramétrés. Format : texte continu, 250 mots max. »

Astuce spécifique – Mentionner *« 5 lignes de raisonnement max »* conserve la profondeur tout en maintenant la latence sous contrôle.

5.2.5 GPT-4.5

Positionnement – le plus « humain » : style naturel, nuance émotionnelle.

Capacités – storytelling, variation de ton, compréhension des intentions implicites.

Pièges – coût élevé, contexte limité à 128 K, légèrement moins fiable sur calculs.

Outils – web et Python disponibles, pas d'appel automatique.

Prompt type

> Définir l'audience, le registre et l'objectif narratif.
>
> Exemple prospection
>
> « Raconte sous forme de mini-scénario (500 mots, ton inspirant) comment une start-up greentech a percé un marché saturé grâce à l'advocacy marketing B2B. Termine par trois leçons actionnables. »

5.2.6 GPT-4.1

Positionnement – modèle API centré développeurs : suivi littéral, long contexte (jusqu'à 1 M token en Nano).

Capacités – génération de diff de code, agents automatiques avec planification, robustesse sur instructions complexes.

Pièges – répond exactement à ce qui est demandé ; toute omission dans le prompt devient une lacune dans la sortie.

Outils – function calling poussé ; meilleurs taux de réussite sur SWE-bench.

Prompt type

> Bloc hiérarchisé : Rôle → Instructions → Exemples (facultatif) → Contexte → Rappel final. **Exemple prospection**

```
# Rôle
Tu es un analyste B2B spécialisé dans les marchés saturés. # Instructions
- Déduis trois métriques clés pour mesurer l'efficacité d'une prospection.
- Propose un plan d'expérimentation A/B pour chacune.
- Formate la sortie en JSON strict { "metrique": "", "plan": "" } # Contexte
(résumé des données internes)
# Rappel
Réponds uniquement en JSON.
```

En appliquant ces profils et ces schémas de prompt, on obtient rapidement le meilleur ratio pertinence / latence / coût pour chaque famille de tâches, sans multiplier les essais empiriques.

6 . CONCLUSION & PROCHAINES ÉTAPES

Le prompt engineering n'est ni un accessoire ni un simple exercice de style : il détermine directement la pertinence, la fiabilité et le délai de génération des réponses d'un modèle. Après avoir défini ses fondements (définition, importance, impacts), nous avons montré que la qualité d'une sortie est indissociable de la maîtrise des hallucinations : précision des consignes, ancrage dans des sources vérifiables et contrôle de la longueur sont les trois leviers clés.

Les sept techniques fondamentales (Chain-of-Thought, self-consistency, injection de documents externes, ReAct, inversion de prompt, décomposition structurée, formats balisés) constituent une boîte à outils suffisante pour traiter la majorité des cas métier, à condition d'être combinées avec un lexique d'instructions clair : verbe d'action, modificateur de qualité, contexte utile.

Le choix du modèle est stratégique : GPT-4o pour la polyvalence, o-series pour le raisonnement profond, GPT-4.5 pour la narration nuancée et GPT-4.1 pour le suivi littéral et le codage. Adapter la structure du prompt à ces profils garantit un bon ratio pertinence/latence. Des astuces ciblées — limiter le contexte sur o3, demander une justification courte sur o4-mini, déclencher la réflexion guidée sur o4-mini-high — optimisent encore le rendement.

Enfin, une démarche d'amélioration continue — tester, mesurer, documenter — est indispensable pour industrialiser la pratique : bibliothèque interne de prompts, revue mensuelle des performances et validation humaine sur les livrables sensibles. À ce prix, le prompt engineering devient un levier opérationnel pour convertir les capacités brutes des LLM d'OpenAI en valeur métier tangible.

6.1 PRINCIPAUX ENSEIGNEMENTS

1. **Le prompt influence directement la qualité et l'utilité de la réponse.**
   Une formulation explicite de l'objectif, du niveau de détail et du format attendu réduit les incohérences et les hallucinations.

2. Chaque modèle OpenAI a un profil distinct.

- GPT-4o pour la polyvalence
- o-series pour le raisonnement détaillé
- GPT-4.5 pour la narration nuancée
- GPT-4.1 pour l'exécution littérale et le codage.

Adapter la structure du prompt à ces spécificités économise du temps et améliore la pertinence.

3. La maîtrise des leviers "Qualité / Hallucination / Latence" repose sur trois axes :

- précision des consignes,
- Ancrage dans des sources vérifiables,
- Contrôle de la longueur pour limiter la génération superflue.

4. Le lexique d'instruction (verbes, modificateurs, contextuels) constitue une trousse de base.

Savoir combiner **action → qualité → contexte** suffit dans la majorité des cas d'usage professionnels.

6.2 BONNES PRATIQUES TRANSVERSES

Étape	Point de vigilance	Recommandation opérationnelle
Conception du prompt	Objectifs implicites	Les expliciter en une phrase avant tout ajout de contexte
Test rapide	Dérive stylistique	Vérifier 2–3 exemples courts avant d'automatiser
Itérations	Accumulation de contexte	Nettoyer régulièrement les chaînes de discussion longues
Déploiement	Modèle non adapté	Maintenir un tableau de correspondance "tâche ↔ modèle"
Revue qualité	Hallucinations ponctuelles	Ajouter une étape humaine de validation sur les livrables critiques

6.3 RESSOURCES COMPLÉMENTAIRES

• **Documentation OpenAI** (developers.openai.com) : description officielle des paramètres par modèle.

• **Prompting Guide (OpenAI Cookbook)** : recueil d'exemples ciblant GPT-4.1 et l'appel de fonctions.

• **Communautés spécialisées** (forums.openai.com, Slack "Prompt Engineering" etc.) : retours d'expérience et modèles de prompts partagés.

PROCHAINES ÉTAPES

1. Sélectionner un cas métier prioritaire et réécrire le prompt à l'aide du lexique de la section 4. 2. Tester ce prompt sur deux modèles différents pour mesurer l'impact latence / qualité.

3. Formaliser les résultats (format, temps de génération, ajustements nécessaires) et enrichir la bibliothèque interne.

Cette démarche itérative, centrée sur l'expérimentation et la capitalisation, permet d'intégrer durablement le prompt engineering dans les processus de production tout en contrôlant les coûts de génération et le temps passé.