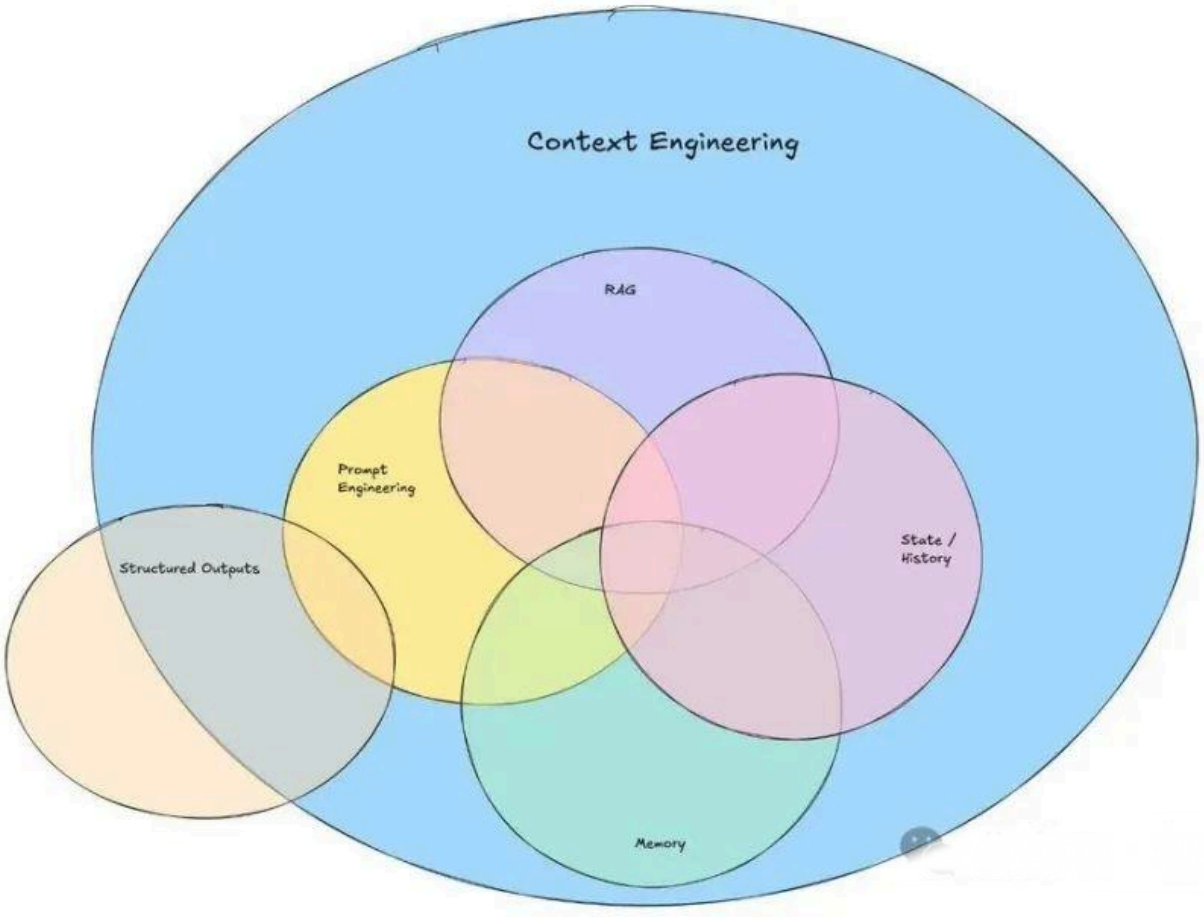
EDUCASIUM

Formations IA

GUIDE D'INGÉNIERIE CONTEXTUELLE

QU'EST-CE QUE L'INGÉNIERIE CONTEXTUELLE ?

Il y a quelques années, de nombreux experts — y compris des chercheurs de haut niveau en IA — prédisaient la disparition imminente de l'ingénierie de prompts.
Force est de constater qu'ils se sont largement trompés : aujourd'hui, cette discipline est plus essentielle que jamais.
Elle est même en train d'être rebaptisée "ingénierie contextuelle", un nouveau terme (certes tendance) pour désigner un processus clé : affiner les instructions et le contexte pertinent dont un LLM a besoin pour accomplir ses tâches de manière efficace.
De nombreux auteurs ont déjà écrit sur le sujet — Ankur Goyal, Walden Yan, Tobi Lütke, Andrej Karpathy — mais j'aimerais ici partager ma propre vision de l'ingénierie contextuelle, et vous proposer un guide concret pas à pas pour la mettre en pratique dans un workflow d'agent IA.
Je ne suis pas certain de l'origine exacte du terme "context engineering", mais nous allons nous appuyer ici sur un schéma proposé par Dex Horthy qui explique brièvement ce concept.
Preguntar a ChatGPT



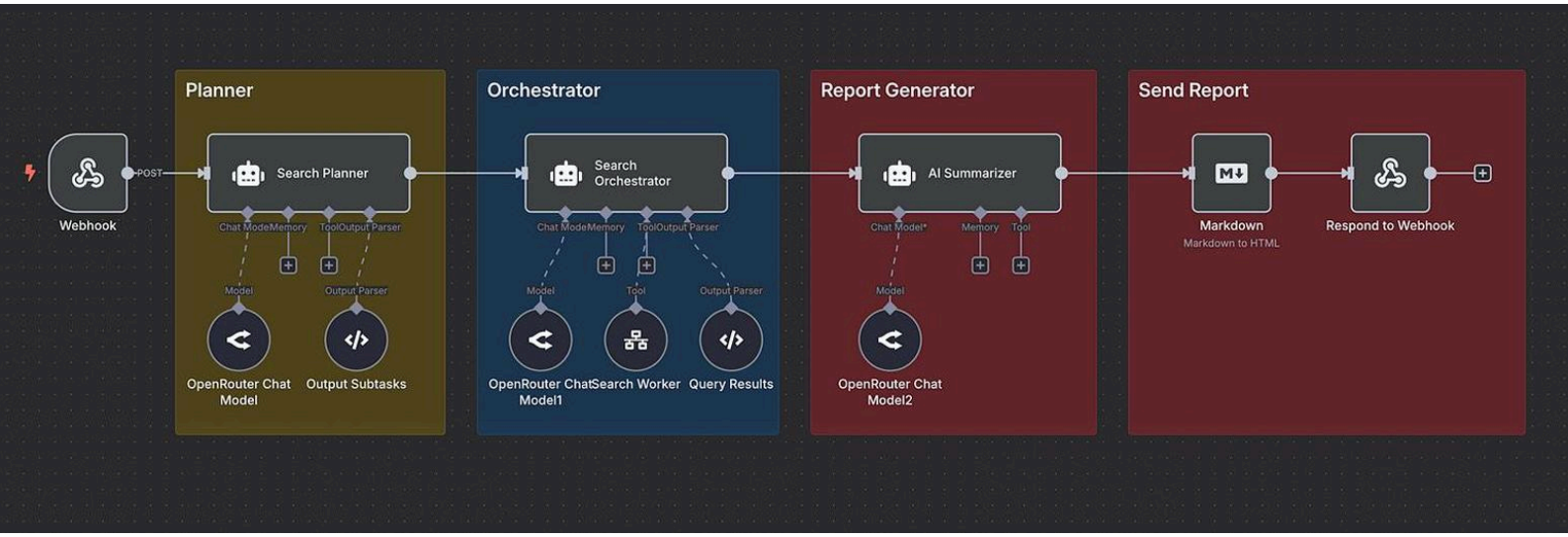
J’aime le terme "ingénierie contextuelle" car il me semble plus large et plus représentatif du travail réel derrière ce qu’on appelle aujourd’hui l’ingénierie de prompts, en y intégrant aussi d’autres tâches connexes.
Le doute sur le sérieux de l’ingénierie de prompts vient souvent du fait qu’on la confond avec le prompting à l’aveugle (c’est-à-dire, simplement poser une question à un modèle comme ChatGPT). Dans le prompting à l’aveugle, on se contente d’interroger le système. En ingénierie de prompts, on réfléchit plus profondément à la structure et au contexte du prompt. Peut-être aurait-il fallu appeler cette discipline ingénierie contextuelle dès le départ.
L’ingénierie contextuelle va plus loin : il s’agit d’architecturer tout le contexte, en dépassant le simple prompt pour adopter des méthodes rigoureuses permettant d’obtenir, enrichir et optimiser les connaissances transmises au système.
Pour un développeur, l’ingénierie contextuelle est un processus itératif visant à optimiser les instructions et le contexte fourni à un LLM pour atteindre un résultat précis. Cela inclut des processus formels d’évaluation (comme les pipelines d’évaluation) pour mesurer l’efficacité des stratégies mises en place.
Étant donné la rapidité d’évolution du domaine de l’IA, je propose une définition élargie de l’ingénierie contextuelle :
Le processus de conception et d’optimisation des instructions et du contexte pertinent destiné aux LLMs et aux modèles d’IA avancés pour qu’ils accomplissent efficacement leurs tâches.
Cela concerne non seulement les LLMs textuels, mais aussi les modèles multimodaux, de plus en plus présents. Cela englobe les efforts liés à l’ingénierie de prompts, mais aussi des processus associés, tels que :

- Conception et gestion de chaînes de prompts (le cas échéant)
- Ajustement des instructions et prompts système
- Gestion des éléments dynamiques du prompt (ex : entrées utilisateur, date/heure)
- Recherche et préparation de connaissances pertinentes (ex : RAG)
- Augmentation de requêtes
- Définition et configuration d’outils (dans le cadre de systèmes agents)
- Préparation et optimisation d’exemples de type few-shot
- Structuration des entrées et sorties (ex : délimiteurs, schéma JSON)
- Gestion de la mémoire à court terme (état et contexte historique) et mémoire à long terme (ex : récupération de connaissances dans un vector store)
- Et bien d’autres astuces permettant d’optimiser les prompts système pour atteindre les objectifs souhaités.

En résumé, le but de l’ingénierie contextuelle est d’optimiser les informations présentes dans la fenêtre de contexte du LLM. Cela implique aussi de filtrer les informations inutiles, ce qui constitue une discipline à part entière nécessitant une mesure systématique de la performance du modèle.
Tout le monde parle d’ingénierie contextuelle. Ici, on va vous montrer un exemple concret de ce que cela signifie lorsqu’on construit des agents IA.

L'INGÉNIERIE CONTEXTUELLE EN ACTION

Prenons un exemple concret d'un travail récent d'ingénierie contextuelle réalisé pour une application de recherche approfondie multi-agents que j'ai développée à titre personnel.
J'ai conçu le workflow agentique dans n8n, mais l'outil utilisé importe peu. Voici à quoi ressemble l'architecture complète des agents que j'ai mise en place :



Dans mon workflow, l'agent Search Planner est chargé de générer un plan de recherche à partir de la requête de l'utilisateur.

EDUCASIUM
Formations IA

PROMPT SYSTÈME

Voici le prompt système que j'ai conçu pour ce sous-agent :

You are an expert research planner. Your task is to break down a complex research query (delimited by) into specific search subtasks, each focusing on a different aspect or source type.

The current date and time is: {{ $now.toISO() }} For each subtask, provide: 1. A unique string ID for the subtask (e.g., 'subtask_1', 'news_update') 2. A specific search query that focuses on one aspect of the main query 3. The source type to search (web, news, academic, specialized) 4. Time period relevance (today, last week, recent, past_year, all_time) 5. Domain focus if applicable (technology, science, health, etc.) 6. Priority level (1-highest to 5-lowest) All fields (id, query, source_type, time_period, domain_focus, priority) are required for each subtask, except time_period and domain_focus which can be null if not applicable. Create 2 subtasks that together will provide comprehensive coverage of the topic. Focus on different aspects, perspectives, or sources of information. Each substask will include the following information:

id: str query: str source_type: str # e.g., "web", "news", "academic", "specialized" time_period: Optional[str] = None # e.g., "today", "last week", "recent", "past_year", "all_time" domain_focus: Optional[str] = None # e.g., "technology", "science", "health" priority: int # 1 (highest) to 5 (lowest) After obtaining the above subtasks information, you will add two extra fields. Those correspond to start_date and end_date. Infer this information given the current date and the time_period selected. start_date and end_date should use the format as in the example below: "start_date": "2024-06-03T06:00:00.000Z", "end_date": "2024-06-11T05:59:59.999Z",

Ce prompt comporte de nombreux éléments qui nécessitent une réflexion approfondie sur le contexte exact à fournir à l'agent de planification pour qu'il accomplisse efficacement sa tâche. Comme vous pouvez le constater, il ne s'agit pas simplement de rédiger une instruction ou un prompt basique : ce processus demande de l'expérimentation et l'intégration d'un contexte pertinent pour que le modèle atteigne un niveau de performance optimal.
Décomposons maintenant le problème en composantes essentielles, qui sont au cœur d'une ingénierie contextuelle efficace.

INSTRUCTIONS

L'instruction correspond aux directives générales fournies au système pour lui indiquer précisément ce qu'il doit faire.

> You are an expert research planner. Your task is to break down a complex research query (delimited by) into specific search subtasks, each focusing on a different aspect or source type.

Beaucoup de débutants — et même certains développeurs IA expérimentés — s'arrêtent à ce stade. Mais comme je l'ai montré avec le prompt complet plus haut, on comprend vite à quel point le système a besoin de bien plus de contexte pour fonctionner comme on le souhaite.
C'est justement ça, l'essence de l'ingénierie contextuelle : fournir au système une compréhension plus fine de la portée du problème et des résultats précis attendus.

ENTRÉE UTILISATEUR

L'entrée utilisateur ne figurait pas dans le prompt système, mais voici un exemple de ce à quoi elle pourrait ressembler.

> What's the latest dev news from OpenAI?

Notez l'utilisation des délimiteurs, qui permet de mieux structurer le prompt. C'est essentiel pour éviter toute confusion et clarifier ce qui constitue l'entrée utilisateur et ce que nous attendons que le système génère.
Parfois, le type d'information fourni en entrée est directement lié à la sortie attendue du modèle (par exemple, une requête principale comme entrée, et des sous-requêtes comme sorties).

ENTRÉES ET SORTIES STRUCTURÉES

En plus de l'instruction générale et de l'entrée utilisateur, vous aurez peut-être remarqué que j'ai accordé une attention particulière aux détails liés aux sous-tâches que l'agent de planification doit générer. Voici ci-dessous les instructions détaillées que j'ai fournies à l'agent afin qu'il puisse créer les sous-tâches à partir de la requête de l'utilisateur.

> For each subtask, provide: 1. A unique string ID for the subtask (e.g., 'subtask_1', 'news_update') 2. A specific search query that focuses on one aspect of the main query 3. The source type to search (web, news, academic, specialized) 4. Time period relevance (today, last week, recent, past_year, all_time) 5. Domain focus if applicable (technology, science, health, etc.) 6. Priority level (1-highest to 5-lowest) All fields (id, query, source_type, time_period, domain_focus, priority) are required for each subtask, except time_period and domain_focus which can be null if not applicable. Create 2 subtasks that together will provide comprehensive coverage of the topic. Focus on different aspects, perspectives, or sources of information.

Si vous observez attentivement les instructions ci-dessus, vous verrez que j'ai choisi de structurer une liste des informations attendues que l'agent de planification doit générer, en y ajoutant des conseils et exemples pour mieux orienter le processus de génération.
C'est essentiel pour offrir à l'agent un contexte supplémentaire sur ce qui est réellement attendu.
Par exemple, si vous ne précisez pas que le niveau de priorité doit être sur une échelle de 1 à 5, le système pourrait par défaut utiliser une échelle de 1 à 10. Ce type de détail contextuel est crucial !
Parlons maintenant des sorties structurées. Pour obtenir des réponses cohérentes de l'agent de planification, nous devons aussi fournir un exemple de format pour les sous-tâches et les types de champs attendus.
Voici ci-dessous l'exemple de sortie que nous transmettons à l'agent comme contexte supplémentaire. Il lui sert d'indice pour comprendre la structure de la réponse souhaitée :
Preguntar a ChatGPT

> Each substask will include the following information: id: str query: str source_type: str # e.g., "web", "news", "academic", "specialized" time_period: Optional[str] = None # e.g., "today", "last week", "recent", "past_year", "all_time" domain_focus: Optional[str] = None # e.g., "technology", "science", "health" priority: int #1 (highest) to 5 (lowest)

En complément, dans n8n, vous pouvez également utiliser un outil d'analyse de sortie (output parser), qui sert à structurer les sorties finales du modèle.
Dans mon cas, j'ai choisi de fournir un exemple au format JSON, comme celui-ci :

{ "subtasks": [{ "id": "openai_latest_news", "query": "latest OpenAI announcements and news", "source_type": "news", "time_period": "recent", "domain_focus": "technology", "priority": 1, "start_date": "2025-06-03T06:00:00.000Z", "end_date": "2025-06-11T05:59:59.999Z" }, { None "id": "openai_official_blog", "query": "OpenAI official blog recent posts", "source_type": "web", "time_period": "recent", "domain_focus": "technology", "priority": 2, "start_date": "2025-06-03T06:00:00.000Z", "end_date": "2025-06-11T05:59:59.999Z" }, ... }

L'outil va ensuite générer automatiquement le schéma à partir de ces exemples, ce qui permet au système de parser et produire des sorties structurées correctement, comme illustré dans l'exemple ci-dessous :

[{ "action": "parse", "response": { "output": { "subtasks": [{ "id": "subtask_1", "query": "OpenAI recent announcements OR news OR updates", "source_type": "news", "time_period": "recent", "domain_focus": "technology", "priority": 1, "start_date": "2025-06-24T16:35:26.901Z", "end_date": "2025-07-01T16:35:26.901Z" }, { "id": "subtask_2", "query": "OpenAI official blog OR press releases", "source_type": "web", "time_period": "recent", "domain_focus": "technology", "priority": 1.2, "start_date": "2025-06-24T16:35:26.901Z", "end_date": "2025-07-01T16:35:26.901Z" }] } } }]

Ça peut paraître complexe, mais aujourd'hui de nombreux outils intègrent nativement des fonctions de sortie structurée, donc il est probable que vous n'ayez pas besoin de tout coder vous-même.
n8n, par exemple, simplifie énormément cette étape de l'ingénierie contextuelle.
C'est d'ailleurs un aspect souvent sous-estimé, que beaucoup de développeurs IA semblent ignorer.
Pourtant, ces techniques sont essentielles.
L'ingénierie contextuelle permet justement de mettre en lumière ces approches puissantes, en particulier lorsque votre agent produit des résultats incohérents et que ces résultats doivent être transmis dans un format spécifique à l'étape suivante du workflow.

OUTILS

Nous utilisons n8n pour construire notre agent, ce qui permet d'intégrer facilement le contexte de la date et de l'heure actuelles.
Voici comment procéder :

> The current date and time is: {{ $now.toISO() }}

Il s'agit ici d'une fonction simple et pratique appelée dans n8n, mais il est courant de créer cela sous forme d'un outil dédié, afin de rendre le système plus dynamique (par exemple, ne récupérer la date et l'heure que si la requête le nécessite).
Et c'est exactement ça, l'ingénierie contextuelle : elle vous oblige, en tant que concepteur, à prendre des décisions concrètes sur quel contexte transmettre au LLM, et à quel moment.
C'est une bonne chose, car cela élimine les hypothèses implicites et réduit les imprécisions dans votre application.
La date et l'heure sont des éléments de contexte importants pour le système : sans eux, il a tendance à mal gérer les requêtes qui dépendent du temps réel.
Par exemple, si je demande au système de chercher les dernières actualités sur le développement chez OpenAI publiées la semaine dernière, il risque de deviner les dates — ce qui mène à des requêtes sous-optimales, donc à des résultats de recherche inexacts.
En fournissant la date et l'heure réelles, le système peut mieux déduire les plages temporelles, ce qui est crucial pour un agent de recherche ou les outils qui en dépendent.
J'ai donc ajouté cette information au contexte transmis, pour permettre au LLM de générer une plage de dates correcte :

> After obtaining the above subtasks information, you will add two extra fields. Those correspond to start_date and end_date. Infer this information given the current date and the time_period selected. start_date and end_date should use the format as in the example below: "start_date": "2024-06-03T06:00:00.000Z", "end_date": "2024-06-11T05:59:59.999Z",

Ici, on se concentre sur l'agent de planification de notre architecture, donc il n'est pas nécessaire d'ajouter trop d'outils à cette étape.
Le seul outil supplémentaire pertinent serait un outil de récupération (retrieval tool), capable de retrouver des sous-tâches pertinentes en fonction d'une requête donnée.
Voyons cette idée un peu plus en détail ci-dessous.

RAG ET MÉMOIRE

La première version de l'application de recherche approfondie que j'ai construite n'utilise pas de mémoire à court terme, mais nous avons développé une variante qui met en cache les sous-requêtes associées à différentes requêtes utilisateur.
C'est très utile pour accélérer le workflow et l'optimiser.
Par exemple, si une requête similaire a déjà été traitée, il est possible de stocker les résultats dans un vector store et de les interroger, afin d'éviter de régénérer un plan déjà existant.
Rappel : chaque appel aux API LLM augmente la latence et les coûts.
C'est là qu'intervient une ingénierie contextuelle intelligente : elle rend votre application plus dynamique, plus économique et plus performante.
Tu vois, l'ingénierie contextuelle ne se limite pas à optimiser ton prompt : c'est aussi le choix stratégique du bon contexte selon les objectifs à atteindre.
Et il y a un fort potentiel créatif dans la manière dont tu gères ton vector store et dont tu réinjectes les sous-tâches déjà générées dans le contexte.
En clair : l'ingénierie contextuelle inventive est ton vrai avantage compétitif.

ÉTATS ET CONTEXTE HISTORIQUE

Ce n'est pas encore implémenté dans la version 1 de notre agent de recherche approfondie, mais une étape clé du projet consiste à optimiser les résultats finaux pour générer un rapport de qualité.
Dans de nombreux cas, un système agentique doit réviser tout ou partie des requêtes, sous-tâches, ou même les données issues des APIs de recherche web.
Cela signifie que le système va itérer plusieurs fois sur le même problème, et pour cela, il a besoin d'accéder aux états précédents et potentiellement à l'historique complet du contexte.
Qu'est-ce que cela implique dans notre cas concret ?
Il s'agit par exemple de donner à l'agent l'accès à :

- l'état des sous-tâches (version en cours, statut, etc.)
- les révisions effectuées
- les résultats précédents produits par chaque agent du workflow
- tout autre contexte pertinent utile à la phase de révision

Ce type de contexte est très dépendant de l'objectif d'optimisation.
C'est là que l'ingénierie contextuelle devient complexe et hautement décisionnelle.
Tu commences à voir le tableau : cette partie demandera de nombreuses itérations.
C'est pourquoi j'insiste autant sur l'importance de l'évaluation.
Si tu ne mesures pas les effets de tes choix contextuels, comment savoir si ton ingénierie fonctionne vraiment ?

INGÉNIERIE CONTEXTUELLE AVANCÉE [EN COURS DE DÉVELOPPEMENT]

Il existe de nombreux autres aspects de l'ingénierie contextuelle que nous n'aborderons pas dans cet article, tels que :

- la compression du contexte
- les techniques de gestion du contexte
- la sécurisation du contexte
- ou encore l'évaluation de son efficacité (c'est-à-dire, mesurer son impact dans le temps)

Nous partagerons bientôt d'autres réflexions sur ces sujets dans de futurs articles.
Le contexte peut se diluer ou devenir inefficace (rempli d'informations obsolètes ou non pertinentes), ce qui exige la mise en place de workflows d'évaluation spécifiques pour détecter ces problèmes.
Je suis convaincu que l'ingénierie contextuelle va continuer d'évoluer pour devenir un ensemble de compétences clés pour les développeurs et ingénieurs IA.
Au-delà de l'ingénierie manuelle, il existe aussi un fort potentiel pour automatiser le traitement du contexte de manière intelligente.
J'ai vu apparaître quelques outils allant dans ce sens, mais le domaine reste encore largement à explorer.

RESSOURCES

Voici quelques lectures recommandées récentes sur l'ingénierie contextuelle :

- Prompting Guide
- Context Engineering by R. Lance Martin
- Tweet de Karpathy sur le sujet
- Article de Phil Schmid
- Simple.ai – The Skill That's Replacing Prompt Engineering
- GitHub – 12 Factor Agents
- LangChain Blog – The Rise of Context Engineering

🔧 Envie d'aller plus loin ?
Je vous recommande mes formations pratiques, dans lesquelles j'aborde en détail toutes ces thématiques avec une approche orientée terrain :