



ICT Training Center

Il tuo partner per la Formazione e la Trasformazione digitale della tua azienda



SPRING AI

GENERATIVE ARTIFICIAL INTELLIGENCE CON JAVA

Simone Scannapieco

Corso base per Venis S.p.A, Venezia, Italia

Settembre 2025

PROMPT ENGINEERING

TECNICHE DI BASE

10:44

ChatGPT >

Mi spieghi cosa è il prompt engineering?

Il **prompt engineering** è l'arte e la tecnica di progettare e formulare input specifici, detti **prompt**, per ottenere risposte desiderate da un modello di intelligenza artificiale (AI) come me. In parole semplici, è il processo di creare domande o richieste ben strutturate per guidare il modello a fornire risultati pertinenti, accurati e utili.

I modelli AI, come quelli basati su **transformer** (ad esempio, GPT), sono progettati per rispondere a input testuali, ma la qualità e la precisione delle loro risposte dipendono molto da come viene formulato il prompt. Ecco alcuni aspetti importanti del prompt engineering:

- Chiarezza e specificità:** Un prompt ben formulato deve essere chiaro e specifico, così che l'AI possa comprendere esattamente cosa gli viene chiesto. Ad esempio, anziché chiedere "Dimmi qualcosa sulla storia", si potrebbe specificare "Quali furono le cause principali della Prima Guerra Mondiale?".

+ Messaggio

10:45

ChatGPT >

- Chiarezza e specificità:** Un prompt ben formulato deve essere chiaro e specifico, così che l'AI possa comprendere esattamente cosa gli viene chiesto. Ad esempio, anziché chiedere "Dimmi qualcosa sulla storia", si potrebbe specificare "Quali furono le cause principali della Prima Guerra Mondiale?".
- Contesto e dettagli:** Fornire contesto o dettagli aggiuntivi all'interno del prompt aiuta il modello a capire meglio ciò di cui hai bisogno. Ad esempio, "Scrivi una breve introduzione sul Rinascimento per un pubblico scolastico".
- Struttura e vincoli:** A volte, può essere utile specificare il formato della risposta, come "Dammi una lista di cinque motivi" o "Scrivi una risposta di massimo 100 parole".
- Uso di esempi:** Se necessario, includere esempi nel prompt può aiutare l'AI a comprendere meglio il tono, lo stile o il tipo di risposta atteso. Ad esempio, "Rispondi come se fossi un personaggio storico, come Leonardo da Vinci".

+ Messaggio

10:45

ChatGPT >

- Contesto e dettagli:** Fornire contesto o dettagli aggiuntivi all'interno del prompt aiuta il modello a capire meglio ciò di cui hai bisogno. Ad esempio, "Scrivi una breve introduzione sul Rinascimento per un pubblico scolastico".
- Struttura e vincoli:** A volte, può essere utile specificare il formato della risposta, come "Dammi una lista di cinque motivi" o "Scrivi una risposta di massimo 100 parole".
- Uso di esempi:** Se necessario, includere esempi nel prompt può aiutare l'AI a comprendere meglio il tono, lo stile o il tipo di risposta atteso. Ad esempio, "Rispondi come se fossi un personaggio storico, come Leonardo da Vinci".
- Iterazioni e perfezionamento:** Il prompt engineering è spesso un processo iterativo. Si potrebbe iniziare con un prompt di base, vedere la risposta del modello, e poi modificarlo o perfezionarlo per ottenere un risultato più utile.

+ Messaggio

➡ Definiamo meglio il concetto di **prompt**. . .

- ➔ Sceglie uno **specifico** modello da ottimizzare
- ➔ Capisce come è stato addestrato il modello
- ➔ Capisce come è stato configurato il modello
- ➔ Comprende come riconfigurare il modello
- ➔ Si esprime efficacemente con il modello
 - ➔ Stile
 - ➔ Tono
 - ➔ Struttura
 - ➔ Scelta delle parole
- ⚠ Spinge il LLM verso un comportamento **pseudo-deterministico**



- 1 **Lunghezza massima:** numero di *token* massimo generato dal modello
 - ➔ Evitare alti consumi (energetici, economici, ...)
 - ➔ Abbattere i tempi di risposta
 - ➔ Cruciale in alcuni tipi di strategie di *prompting* (ReAct) e per alcune tipologie di *task*
 - ⚠ **Non** impone modifiche stilistiche al modello!

2 Creatività del modello: libertà nella scelta del *token* successivo

➔ Temperatura:

➔ **Bassa** se vogliamo risposte più deterministiche

➔ **Alta** per risposte "creative"

⚠ Intervallo $[0, +\infty]$

➔ Top-K: seleziona i primi K *token* più probabili dalla distribuzione

⚠ Intervallo $[0, +\infty]$

➔ Top-P: seleziona i *token* più probabili e la cui probabilità cumulativa non supera P

⚠ Intervallo $[0, 1]$

⚠ Non sempre tutti disponibili!

➔ Se tutte disponibili, Top-K e Top-P scremano, temperatura decide

➔ Se temperatura manca, Top-K e Top-P scremano, processo *random* decide

➔ Attenzione alle configurazioni *borderline*

Setup	Effetto
Temp = 0 Temp \gg 0	Top-P, Top-K irrilevanti (<i>greedy decoding</i>) Temp irrilevante
Top-K = 1 Top-K \gg 1	Temp, Top-P irrilevanti (<i>greedy decoding</i>) Decide Temp ma processo decisamente <i>random</i>
Top-P = 0 Top-P = 1	Temp, Top-K irrilevanti (per maggioranza dei modelli) Top-P irrilevante



Yzma e Kronk

©Disney

➔ Per iniziare (secondo Google)

Obiettivo	Temp	Top-P	Top-K
Risposte formali	.2	.95	30
Risposte creative	.1	.9	20
Risposte molto creative	.9	.99	40
Risposta matematica	0	—	—

- 3 Sequenze di terminazione: bloccano la terminazione della generazione
 - Generazione di una lista di non più di 10 punti \leadsto `stop_sequence=[11]`
- 4 Penalità di ripetizione: incentivare il modello a sfruttare la ricchezza del suo dizionario
 - *Frequency penalty*: penalità calcolata su ciascun *token* proporzionale al numero di apparizioni nella sequenza generata
 - *Presence penalty*: penalità **globale** indipendente da *token* o numero di occorrenze

Direttive istruzioni specifiche relative alla *task* da eseguire

Esempi per guidare il LLM verso un *output* più efficace

Ruolo (persona) la prospettiva che il LLM deve adottare per definire tono, stile e contenuto della risposta

“Contesto” informazioni aggiuntive che possono indirizzare e addestrare il modello in modo che generi risposte migliori, pertinenti e coerenti con l’obiettivo

Dati di *input* il testo contenente l’istanza della *task* da risolvere

Indicatore di *output* la tipologia o il formato che si vuole ottenere nella risposta

➡ Dati di *input* obbligatori

➡ Altri dati opzionali

➔ Direttive

- ⚠ Essere chiari e concisi
- ⚠ Evitare istruzioni ambigue e vaghe
- ⚠ Quando possibile, utilizzare verbi che descrivono le azioni da eseguire (“Traduci”, “Fai una lista di ...”)

➔ Esempi

- ⚠ Chiari e rilevanti per la *task*
- ⚠ Mostrare la struttura o contenuto atteso
- ⚠ Numero di esempi commisurato alla complessità della *task*

➔ Ruolo (persona)

- ⚠ Aggiungere *expertise* o una prospettiva specifica
- ⚠ Ruolo allineato alla *task* (es., esperto di *marketing* per generare contenuto promozionale)
- ⚠ Combinare ruolo con il “contesto”

➔ Contesto

- ⚠ Includere informazione strettamente rilevante
- ⚠ Informazione strettamente collegata alla *task*
- ⚠ Fornire un *background*, soprattutto in sotto-casi specifici

➔ Indicatore di *output*

- ⚠ Se possibile, sfruttare meccanismi automatici ottimizzati (es. `BeanOutputConverter` in Spring AI)

- 1 Esempi (se necessari)
 - 2 “Contesto” (informazioni aggiuntive)
 - 3 Ruolo (persona)
 - 4 Direttive
 - 5 Dati di *input*
 - 6 Indicatore di *output*
- ⚠ Evitare che LLM generi **ulteriore contesto** prima di seguire le istruzioni

Implicit solo dati di *input*, senza nemmeno definire le direttive

- ➡ Sfruttare le capacità emergenti del LLM

0-Shot *task* e dati di *input* specificati, ma priva di esempi

Few-shots la richiesta dell'utente è seguita da alcuni esempi su cui la LLM tara il *template* delle risposte da fornire

Role-based le informazioni del *prompt* sono suddivise tra una entità *superuser* (*system* o *admin*) e una entità *user*

- ➡ Esempi, contesto, direttive e indicatore *output* al *system*
- ➡ Dati di *input* allo *user*

Step-back suddividere la richiesta finale al LLM in due fasi

- ➡ Porre una domanda generica pertinente alla *task* finale
- ➡ Incapsulare la risposta in un nuovo *prompt* per modellare la *task*
- ⚠ Attivare dei **percorsi neurali** e **processi di ragionamento**

- ➔ **Fornire esempi:** strumento di insegnamento sempre efficace per allineare il LLM al nostro risultato atteso
- ➔ **Mirare alla semplicità:** utilizzare formulazioni ad alta entropia informativa
 - ➔ Se la formulazione è confusa ed ambigua già per l'utente, ...
 - ➔ Linguaggio semplice
 - ➔ Evitare informazione inutile ("Potresti dirmi quale..." ⇒ "Quale...")
 - ➔ Verbi che descrivono azioni ("Fai una lista...di" ⇒ "Elenca...")
- ➔ **Istruzioni piuttosto che vincoli:** dire "cosa fare" piuttosto che "non fare..."
 - ➔ Evitare che il LLM debba chiedersi cosa è permesso generare se messo davanti ad una lista di vincoli
 - ➔ Gli LLM tendono a trovare scappatoie (ciò che non è esplicitamente vietato è permesso...)
- ➔ **Mai sottovalutare le tecniche più semplici:** possono riservare piacevoli sorprese...
 - ➔ Tenersi sempre aggiornati sulle capacità emergenti dei modelli

- ➔ Sperimentare con formati di *input* e stili di scrittura
 - ➔ In caso di utilizzo di LLM in ambiti non prettamente *business*
 - ➔ Wave del *JSON prompting* e protocolli proprietari (es. OpenAI Harmony)
- ➔ Adattamento agli *update* del modello
 - ➔ Modellare lo stile di *prompting* in maniera retro-attiva
- ➔ Affidarsi ad *output* fortemente strutturati
 - ➔ Spring AI si sta muovendo nella giusta direzione...

- ➔ Documentazione dei tentativi di *prompting*
- ➔ Piattaforme specializzate (MLFlow)
- ➔ Sperimentazione con altri *prompt engineers*
- ➔ LLM istruita su informazioni create da miliardi di persone

*Nessun uomo è un'isola, completo in se stesso;
ogni uomo è un pezzo del continente, una parte
del tutto.*

Thomas Merton, 1956