Προχωρημένη χρήση LLMs

Ηρακλής Βαρλάμης

Στόχοι ενότητας

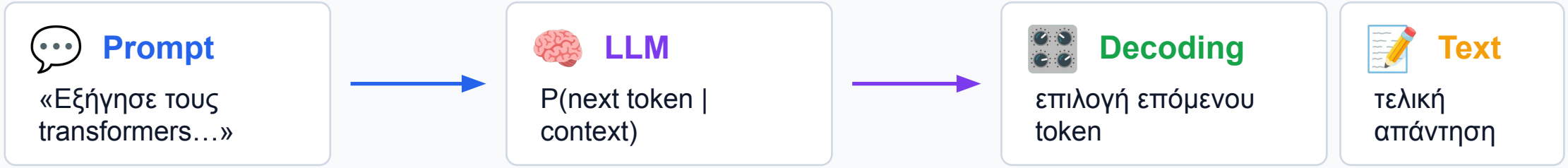
Το LLM δεν είναι μόνο μοντέλο. Είναι σύστημα: decoding + prompt + knowledge + tools + evaluation.

Μέρος Α

Α. Παράμετροι παραγωγής

Από τις πιθανότητες του επόμενου token στις πρακτικές επιλογές decoding.

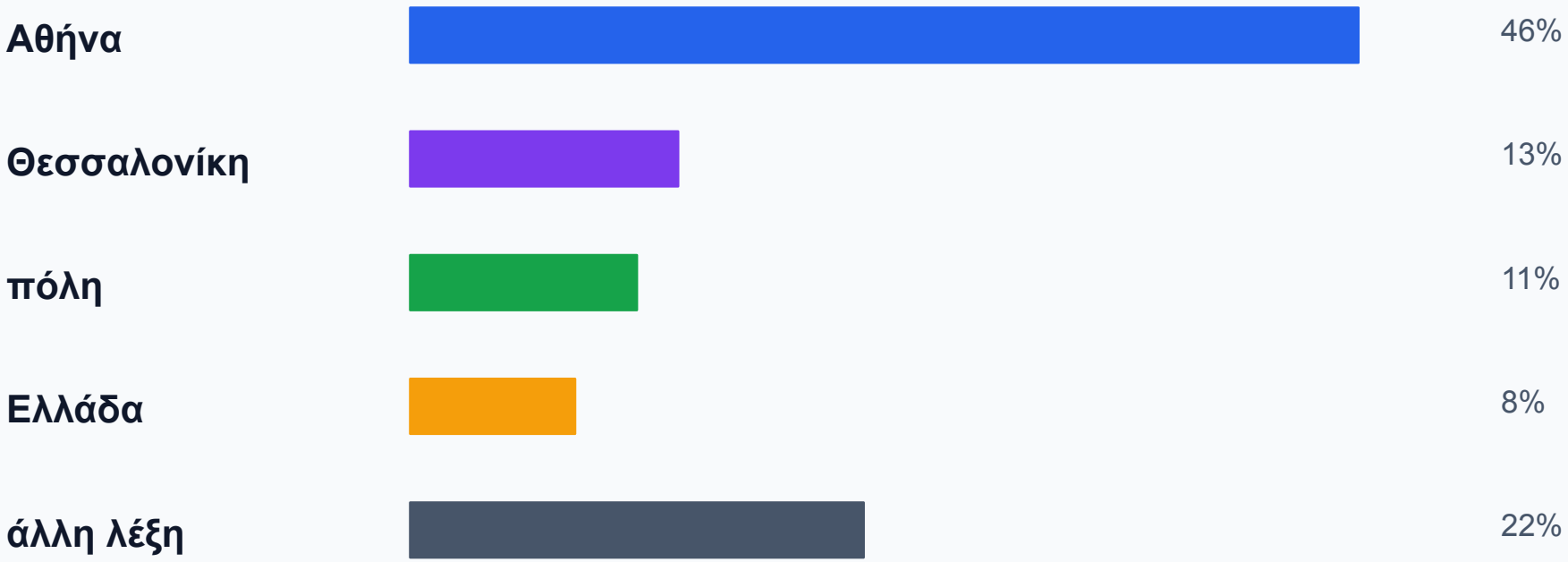
Από πρόβλεψη token σε παραγωγή



Το μοντέλο δίνει κατανομή πιθανοτήτων. Το decoding αποφασίζει πώς μετατρέπεται αυτή η κατανομή σε κείμενο.

Η κατανομή πιθανοτήτων πάνω στο λεξιλόγιο

Παράδειγμα: «Η πρωτεύουσα της Ελλάδας είναι …»



⚠️ **Μήνυμα**

Η «γνώση» του μοντέλου και η «στρατηγική επιλογής» δεν είναι το ίδιο.

Greedy decoding: πάντα το πιο πιθανό token

🏆 **Κανόνας**

Σε κάθε βήμα επιλέγουμε το token με τη μέγιστη πιθανότητα.

✅ **Πλεονέκτημα**

Ντετερμινιστικό και γρήγορο.

⚠️ **Μειονέκτημα**

Μπορεί να γίνει επίπεδο ή επαναληπτικό.

Κατάλληλο για:

- εξαγωγή πεδίων
- κώδικα με αυστηρό format
- σύντομη τεκμηριωμένη απάντηση

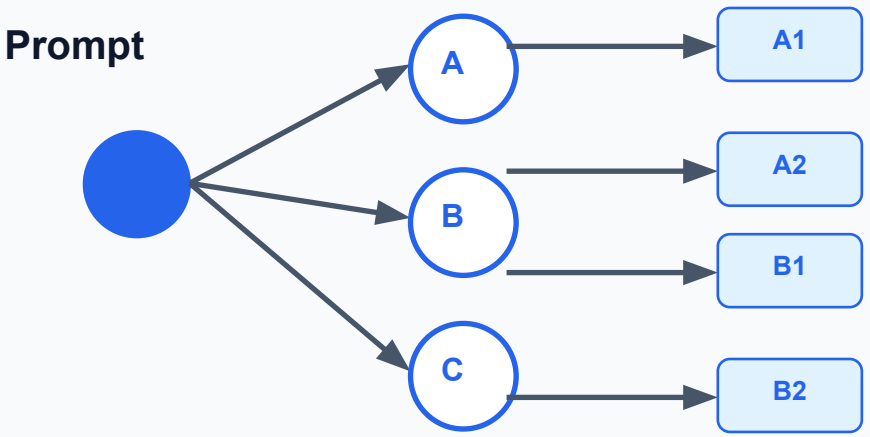
Όχι ιδανικό για:

- δημιουργική γραφή
- brainstorming
- πολλαπλές εναλλακτικές

Beam search: κρατάμε πολλές υποψήφιες συνέχειες

🌳 **Ιδέα**

Κρατάμε τα καλύτερα beam candidates.



🔎 **Πότε βοηθά;**

Μετάφραση, περίληψη και constrained generation. Σε ανοιχτό διάλογο μπορεί να παράγει υπερβολικά «ασφαλές» ή στερεότυπο κείμενο.

Sampling: δειγματοληψία από την κατανομή

🎲 Ιδέα
Δεν επιλέγουμε πάντα το top token. Δειγματοληπτούμε με βάση τις πιθανότητες.

✨ Κέρδος
Περισσότερη ποικιλία, λιγότερο μηχανικό ύφος.

⚠️ Κίνδυνος
Πιο πιθανές ασυνέπειες ή hallucinations.

Το sampling είναι χρήσιμο όταν θέλουμε πολλαπλές εκδοχές ή δημιουργική εξερεύνηση.

Παράδειγμα χρήσης: «Δώσε 5 διαφορετικές αναλογίες για να εξηγήσω το attention.»

Temperature: πόσο «αιχμηρή» είναι η κατανομή;

❄️ Χαμηλή θερμοκρασία

Πιο προβλέψιμες και σταθερές απαντήσεις. Καλή για QA, extraction, κώδικα.

⚖️ Μεσαία

Ισορροπία ανάμεσα σε σταθερότητα και ποικιλία.

🔥 Υψηλή

Πιο δημιουργικές απαντήσεις, αλλά μεγαλύτερο ρίσκο λαθών.

Κανόνας διδασκαλίας

Όσο πιο κρίσιμη είναι η ακρίβεια, τόσο χαμηλότερα κρατάμε τη θερμοκρασία.

Top-k και Top-p / nucleus sampling

🔢 Top-k

Κρατάμε τα k πιθανότερα tokens και αγνοούμε όλα τα υπόλοιπα. Απλό και ελεγχόμενο, αλλά λιγότερο προσαρμοστικό.

🧪 Top-p

Κρατάμε το μικρότερο σύνολο tokens που αθροίζει πιθανότητα p. Προσαρμόζεται στη μορφή της κατανομής.

Top-k = σταθερός αριθμός επιλογών

Top-p = σταθερή μάζα πιθανότητας

Στην πράξη συχνά χρησιμοποιούμε temperature + top-p μαζί.

Demo: ίδιο prompt, διαφορετικό decoding

📌 Factual QA
temperature 0–0.3
σύντομη απάντηση
stop sequences

💡 Brainstorming
temperature 0.8–1.2
top-p 0.9–0.95
πολλαπλές εκδοχές

🧾 Περίληψη
beam ή low-temp
max_new_tokens
anti-repetition

💻 Code
χαμηλή temp
strict format

Prompt demo:
«Εξήγησε τους transformers σε φοιτητή 1ου έτους σε 80 λέξεις.»

Στόχος: οι φοιτητές να δουν ότι οι παράμετροι decoding αλλάζουν ύφος, ποικιλία και αξιοπιστία.

Μέρος Β

Β. Prompting, RAG ή Fine-tuning;

Πότε αλλάζουμε την είσοδο, πότε προσθέτουμε γνώση και πότε εκπαιδεύουμε το μοντέλο.

Το πρόβλημα προσαρμογής

Έχουμε ένα γενικό LLM. Πώς το κάνουμε χρήσιμο για τη δική μας εργασία;

✍️ **Prompting**
οδηγίες και παραδείγματα στο prompt

📚 **RAG**
εξωτερική γνώση τη στιγμή της απάντησης

🔧 **Fine-tuning**
μάθηση συμπεριφοράς στα βάρη

🛠️ **Tools**
ενέργειες σε APIs και συστήματα

📏 **Evals**
μετράμε πριν αποφασίσουμε

Η σωστή επιλογή εξαρτάται από το αν λείπει γνώση, συμπεριφορά, μορφή ή δράση.

Prompt engineering: αλλάζουμε τις οδηγίες, όχι το μοντέλο

✅ Καλό για

ύφος, μορφή, role, few-shot παραδείγματα, γρήγορα πειράματα

⚠️ Όρια

δεν προσθέτει μόνιμη γνώση, μεγαλώνει το prompt, γίνεται εύθραυστο

🚦 Πρακτικά

ξεκινάμε πάντα από prompting πριν πάμε σε πιο ακριβές λύσεις

Pattern

Ρόλος + Εργασία + Περιορισμοί + Παραδείγματα + Μορφή εξόδου

Few-shot prompting: διδάσκουμε με παραδείγματα

Classify the review as Positive or Negative.

Review: «Το φαγητό ήταν εξαιρετικό.»
Label: Positive

Review: «Η παράδοση άργησε πολύ.»
Label: Negative

Review: «Η εφαρμογή κολλάει συχνά.»
Label:

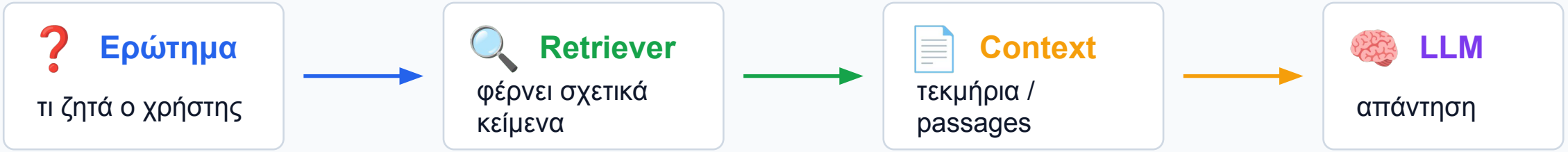
Τι μαθαίνει το μοντέλο;

Μορφή απάντησης, κατηγορίες, ύφος και επίπεδο λεπτομέρειας.

Πότε δεν αρκεί;

Όταν χρειάζεται πολύ ειδική συμπεριφορά ή πολλά παραδείγματα που δεν χωρούν στο context.

RAG: προσθέτουμε γνώση τη στιγμή της απάντησης



Καλό για:

ιδιωτικά έγγραφα

συχνά μεταβαλλόμενη γνώση

απαντήσεις με πηγές

μείωση hallucinations πάνω σε corpus

Δεν λύνει αυτόματα:

κακό chunking

λάθος retrieval

αντιφατικά έγγραφα

prompt injection μέσα στα έγγραφα

Fine-tuning: αλλάζουμε τη συμπεριφορά του μοντέλου

📦 Τι δίνουμε

Παραδείγματα εισόδου και επιθυμητής εξόδου.

✅ Τι κερδίζουμε

Σταθερότερο ύφος, format και task behavior.

⚠️ Τι δεν κάνει καλά

Δεν είναι ο καλύτερος τρόπος για προσθήκη νέων/φρέσκων facts.

Παράδειγμα καλού fine-tuning task

«Για κάθε email υποστήριξης, επέστρεψε JSON με intent, urgency, sentiment και προτεινόμενη απάντηση.»

Το fine-tuning είναι επένδυση. Χρειάζεται dataset, μετρικές και σύγκριση με baseline.

Evals first: χωρίς μέτρηση, δεν ξέρουμε τι βελτιώθηκε

Fine-tuning χωρίς αξιολόγηση = ακριβό guessing.

① 1. Baseline

τρέχον μοντέλο + καλό prompt

② 2. Evaluation set

πραγματικά παραδείγματα + αναμενόμενη έξοδος

③ 3. Compare

accuracy, cost, latency, failure modes

Μόνο αν το fine-tuned μοντέλο κερδίζει καθαρά, αξίζει η πολυπλοκότητα.

RAG vs Fine-tuning: η βασική διάκριση

RAG = γνώση στο context

📚 Χρησιμοποίησε RAG όταν…

- λείπουν facts
- χρειάζονται πηγές
- η γνώση αλλάζει συχνά
- τα δεδομένα είναι ιδιωτικά ή μεγάλα

Fine-tuning = συμπεριφορά στα βάρη

🔧 Χρησιμοποίησε fine-tuning όταν…

- η εργασία είναι σταθερή
- θες σταθερό format
- έχεις πολλά καλά παραδείγματα
- το prompt έγινε πολύ μεγάλο/εύθραυστο

Συχνά η καλύτερη λύση είναι συνδυασμός: RAG για γνώση + fine-tuning για συμπεριφορά.

Decision tree για επιλογή στρατηγικής

1 **Λείπει γνώση;**
Ναι → RAG

→

2 **Λείπει format/ύφος;**
Prompt → Fine-tune

→

3 **Χρειάζεται δράση;**
Tools / Agent

📄 **Κανονισμοί σχολής**
RAG με citations

🏷️ **Ταξινόμηση ειδήσεων**
fine-tune ή encoder classifier

🌦️ **Ζωντανός καιρός**
tool/API call

🎧 **Στυλ support**
prompt, μετά Supervised FT

Άσκηση: για κάθε σενάριο, οι φοιτητές αιτιολογούν επιλογή και failure mode.

Προαιρετικά: preference / reinforcement fine-tuning

Ιδέα

Δεν έχουμε πάντα μία “σωστή” απάντηση. Μπορούμε να βαθμολογούμε απαντήσεις με grader ή προτιμήσεις.

Κατάλληλο για

reasoning tasks, ranking, πολύπλοκη ποιότητα, domain rubrics

Προσοχή

πιο ακριβό, πιο δύσκολο να αξιολογηθεί, χρειάζεται καλό reward signal

Supervised Fine Tuning based on RL

Παράγουμε περισσότερες εναλλακτικές απαντήσεις με χρήση διαφορετικών μοντέλων ή παραμέτρων.
Ο χρήστης βαθμολογεί την απάντηση που παίρνει (από κάποια από τις εναλλακτικές).
Η ανατροφοδότηση του χρήστη χρησιμοποιείται για να προτιμηθεί/αγνοηθεί το μοντέλο την επόμενη φορά.

Μέρος C

C. Agents και Tool Use

Από απλή απάντηση σε ελεγχόμενες ενέργειες με εργαλεία, κατάσταση και αξιολόγηση.

Από LLM σε agent

🧠 LLM

Παίρνει prompt και παράγει απάντηση.

→

🤖 Agent

Σχεδιάζει βήματα, καλεί εργαλεία, παρατηρεί αποτελέσματα και συνεχίζει.

Η διαφορά δεν είναι μόνο το μοντέλο. Είναι ο βρόχος ελέγχου γύρω από το μοντέλο.

agent = LLM + tools + state + policy + safeguards

Tool calling: το μοντέλο ζητά εξωτερική συνάρτηση

🧮 **Calculator**

ακριβείς υπολογισμοί

🔍 **Retriever**

αναζήτηση σε docs

🗄️ **Database**

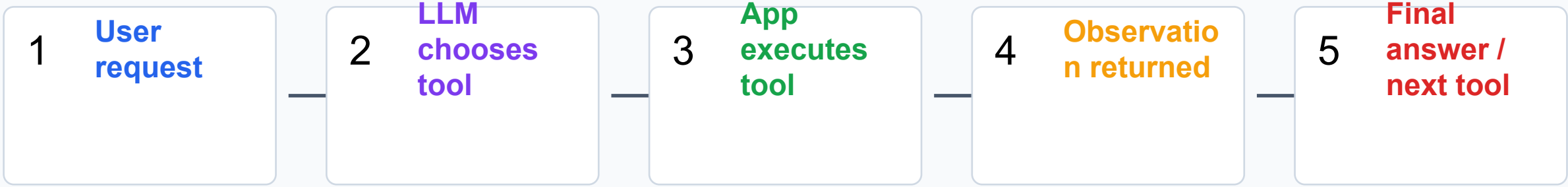
SQL / queries

🌐 **API**

καιρός, emails, tickets

Το LLM δεν εκτελεί μαγικά. Προτείνει tool call με arguments. Η εφαρμογή εκτελεί και επιστρέφει αποτέλεσμα.

Ο βρόχος tool calling



Σημαντικό: το εργαλείο εκτελείται από τον κώδικα της εφαρμογής, όχι από το LLM ανεξέλεγκτα.

Tool schema: το συμβόλαιο μεταξύ LLM και εφαρμογής

```
def search_documents(
    query: str,
    k: int = 5
) -> list[Document]:
    """Return top-k passages
    relevant to the query."""
```

✅ **Καλό schema**

σαφές όνομα, τύποι, περιγραφές, περιορισμοί

🚫 **Κακό schema**

ασαφή arguments, υπερβολικά γενικό εργαλείο, ανεξέλεγκτη πρόσβαση

Έλεγχος επιλογής εργαλείων

🤖 Automatic

το μοντέλο αποφασίζει αν και ποιο tool θα καλέσει

📌 Required

πρέπει να χρησιμοποιηθεί κάποιο tool

🔒 Forced

επιβάλλουμε συγκεκριμένο tool

Σε παραγωγικά συστήματα δεν δίνουμε απεριόριστη ελευθερία.

Χρειαζόμαστε allow-lists, validation, user confirmation για κρίσιμες ενέργειες και logging.

Chains vs Agents

⛓️ Chain

Σταθερή ακολουθία βημάτων.

Πιο προβλέψιμη, πιο εύκολη στο debugging, ιδανική για κλασικό RAG QA.

🤖 Agent

Δυναμική επιλογή βημάτων.

Πιο ευέλικτος, αλλά δυσκολότερος στην αξιολόγηση και πιο ριψοκίνδυνος.

Κανόνας: χρησιμοποίησε chain όταν ξέρεις τη διαδικασία. Χρησιμοποίησε agent όταν η διαδικασία πρέπει να αποφασίζεται δυναμικά.

Memory και state: τι θυμάται ο agent;

💬 Conversation

ιστορικό διαλόγου και προτιμήσεις χρήστη

📋 Task state

στόχος, ενδιάμεσα βήματα, εκκρεμότητες

📝 Scratchpad

σκέψεις/observations που χρειάζονται για εκτέλεση

🧠 Long-term

εξωτερική μνήμη, vector store, database

Η μνήμη αυξάνει τη χρησιμότητα αλλά και τον κίνδυνο: λάθος state, privacy leakage, παλιά δεδομένα.

Multi-agent patterns: χρήσιμα, αλλά όχι δωρεάν

🗺️ Planner

σπάει το πρόβλημα σε βήματα

🔍 Researcher

βρίσκει πηγές και στοιχεία

💻 Coder

γράφει ή εκτελεί κώδικα

🧐 Critic

ελέγχει ποιότητα/λάθη

⚙️ Executor

κάνει action

Πλεονέκτημα: διαχωρισμός ρόλων. Μειονέκτημα: κόστος, latency, debugging, αξιολόγηση.

Ξεκινάμε απλά: single chain → single agent → multi-agent μόνο αν υπάρχει πραγματική ανάγκη.