

LECTIA 2

Versiunea 5.0 - 11 iulie 2026

Cum apare un răspuns din numere

De la mesaj la spațiul latent și răspunsul construit token cu token



Viorel Spînu AI Enabler

curs.viorelspinu.com

Înainte să începem

Pune-i unui model două întrebări. Prima să fie dintr-un domeniu pe care îl cunoști foarte bine. A doua să fie despre ceva la care nu te pricepi deloc.

Cele două răspunsuri vor veni cu aceeași voce calmă. Vor avea propoziții curate, explicații ordonate și concluzii spuse fără bâlbâială. La prima întrebare vezi imediat unde a greșit. La a doua, fluentă te poate face să iei răspunsul drept dovadă de inteligență.

În lecția 1 am separat modelul de aplicație și de agent. Acum deschidem motorul. Urmărim mesajul tău până când apare primul token al răspunsului, apoi vedem de ce aceeași mașinărie poate scrie o explicație bună și o invenție convingătoare.

Cuprins

01. De la mesaj la primul token

Tokenii sunt bucățile pe care le procesează modelul

02. Cum ajung tiparele limbajului în spațiul numeric

Parametrii păstrează tiparele învățate

Cum transformă antrenarea numerele în tipare de limbaj

Geometria poate păstra relații din limbaj

03. Răspunsul se construiește din mers

Fiecare token schimbă continuarea

Promptul intră în textul pe care modelul îl continuă

04. De ce un răspuns fluent poate conține informații false

Creierul nostru confundă ușor fluența cu inteligența

Modelele ghicesc când informația lipsește

Tonul sigur și incertitudinea sunt lucruri diferite

Cum împinge feedbackul modelul să aprobe utilizatorul

05. Contextul și uneltele schimbă rezultatul

Ce informații din context schimbă răspunsul

Uneltele aduc informație și pot executa calcule

06. Lecția 2 într-o pagină

07. Ce urmează

Două experimente până la lecția următoare

De la mesaj la primul token

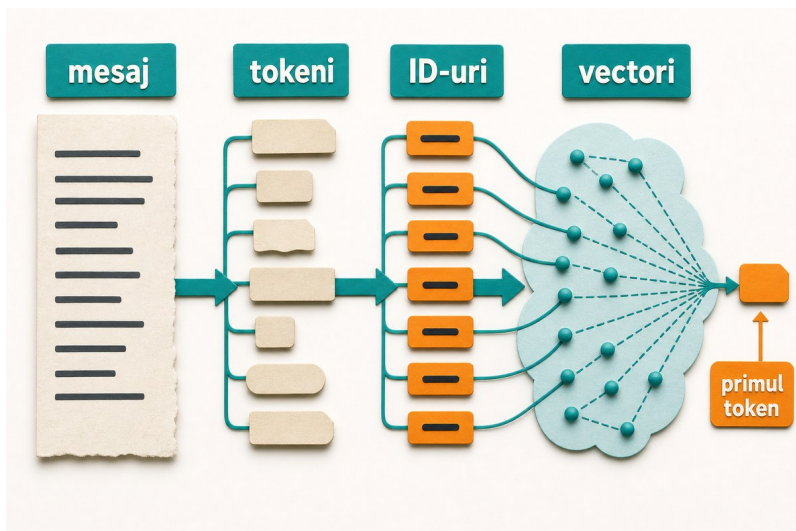
Scrit: „Pregătește un email prin care anunț că nu ajung azi la birou.”
Tu vezi o propoziție. Modelul primește o secvență pe care o poate transforma în calcule.

Tokenii sunt bucățile pe care le procesează modelul

Primul pas se numește tokenizare. Un token poate fi un cuvânt scurt, o bucată de cuvânt, un spațiu sau un semn. Cuvântul **îmbunătățire** poate ajunge în mai multe fragmente, în funcție de tokenizerul folosit.

Fiecare token primește un ID numeric. ID-ul funcționează ca un număr de inventar. Numărul **52332** nu înseamnă singur „redactează”, la fel cum numărul unei cărți din bibliotecă nu conține povestea cărții.

Modelul caută apoi lista de numere asociată acelui ID, numită vector sau embedding. Pe măsură ce vectorul trece prin rețea, se schimbă în funcție de celelalte fragmente din mesaj. **Bancă** ajunge să fie tratat diferit în „stau pe bancă” și în „merg la bancă”. Contextul îi schimbă reprezentarea.



Același mesaj trece prin tokeni, ID-uri și reprezentări numerice înainte să apară primul token

Imaginea separă trei lucruri care se amestecă ușor: fragmentul de text, ID-ul lui și reprezentarea numerică folosită în calcul. Sensul apare din relațiile dintre multe numere și din contextul propoziției.

Modelul desface mesajul în tokeni și îi transformă în reprezentări numerice dependente de context. ID-ul unui token indică poziția lui în vocabularul modelului. Reprezentarea folosită în calcul se formează din vectori și din relațiile cu fragmentele din jur.

Cum ajung tiparele limbajului în spațiul numeric

Avem două lumi. În prima sunt cuvinte, scene, obiecte și relații pe care oamenii le recunosc. În a doua sunt vectori, parametri și transformări matematice. Modelul are nevoie de o trecere între ele.

Parametrii păstrează tiparele învățate

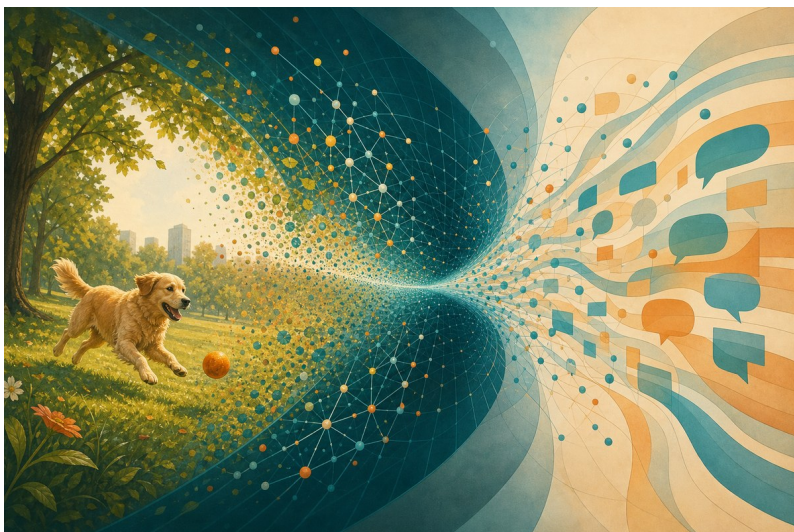
Un LLM este o colecție uriașă de numere organizate într-o rețea. Numerele acestea se numesc parametri. La început produc aproape numai zgomot. Pe durata antrenării sunt ajustate iar și iar.

Modelul primește, de exemplu, începutul „Am intrat în cafenea și am comandat...”. Propune o continuare. Textul de antrenare conține continuarea folosită drept țintă. Calculatorul măsoară diferența și schimbă puțin parametrii. După foarte multe astfel de corecții, „o cafea” ajunge mai aproape de contextul cafenelei decât „un calorifer”.

Corecțiile schimbă valori numerice. Repetate de foarte multe ori, ele ajută rețeaua să învețe acordul dintre cuvinte, structura unei explicații, relațiile dintre concepte și tonurile folosite în diferite situații.

Cum transformă antrenarea numerele în tipare de limbaj

Imagineză-ți modelul ca pe o gaură de vierme între realitate și matematică. De o parte este lumea descrisă prin limbaj. De cealaltă sunt numerele. Prin antrenare, urmele realității din texte devin o hartă numerică a înțeleșurilor: conceptele întâlnite în contexte asemănătoare ajung în regiuni apropiate. Când modelul răspunde, relațiile de pe hartă redevin tokeni.



Gaura de vierme face trecerea dintre limbajul oamenilor și spațiul latent numeric

Gaura de vierme nu apare dintr-odată. Fiecare predicție și fiecare corecție ajustează puțin maparea dintre limbaj și numere. După miliarde de exemple, modelul poate primi o propoziție pe care nu a mai văzut-o și poate calcula o continuare care se potrivește.

Spațiul acesta numeric este numit adesea spațiu latent. „Latent” înseamnă că relațiile există în reprezentările interne și devin vizibile

prin ce poate face modelul. Tiparul pentru **câine** sau **aleargă** este distribuit în multe numere, fără un sertar ori un parametru dedicat unui singur sens.

Geometria poate păstra relații din limbaj

În articolul „Linguistic Regularities in Continuous Space Word Representations”, publicat în 2013, Tomas Mikolov, Wen-tau Yih și Geoffrey Zweig au arătat un exemplu devenit celebru. În reprezentările lor, operația **rege – bărbat + femeie** ajungea în apropierea vectorului pentru **regină**.

Exemplul vine dintr-o generație mai veche de reprezentări, dar face vizibilă ideea: o relație exprimată prin cuvinte poate apărea ca o direcție între vectori, iar calculatorul poate face operații cu acești vectori. Modelele actuale duc această geometrie mult mai departe și schimbă reprezentarea unui cuvânt după propoziția în care apare.

Gaura de vierme explică felul în care modelul lucrează cu sensuri prin reprezentări distribuite. Prin antrenare, rețeaua a învățat să reprezinte tiparele limbajului sub formă numerică și să facă operații cu ele.

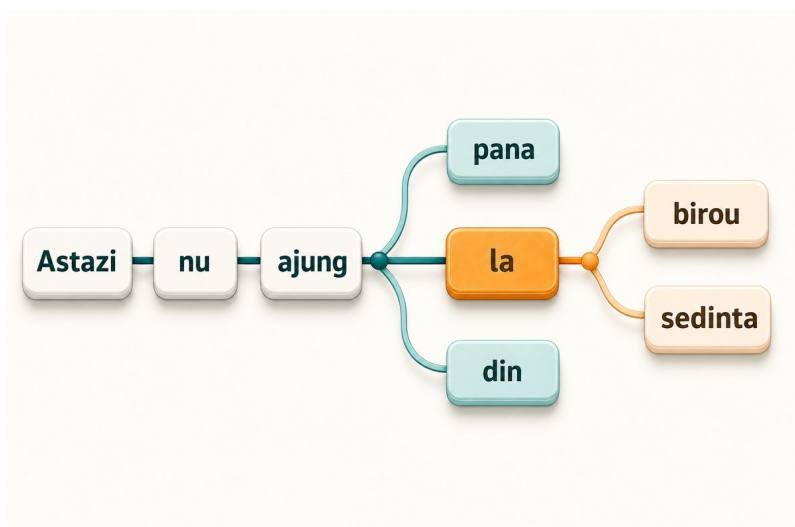
Antrenarea ajustează parametrii prin foarte multe predicții și corecții, până când rețeaua poate reprezenta numeric tipare din limbaj. În spațiul latent, relațiile dintre concepte sunt răspândite prin mulți parametri. Modelul nu păstrează câte un sertar cu definiții sau copii exacte ale textelor văzute, ci folosește relațiile distribuite în parametri pentru a construi continuarea.

Răspunsul se construiește din mers

După ce procesează mesajul, modelul produce scoruri pentru tokenii care ar putea urma. La începutul unui email, **Bună**, **Salut** sau **Stimate** pot apărea printre variantele cu scor mare. Sistemul selectează un token, îl adaugă textului și repetă calculul.

Fiecare token schimbă continuarea

Să pornim cu **Astăzi nu ajung**. Modelul poate continua cu **la**, **până**, **din** sau un semn de punctuație. Dacă apare **la**, următoarele variante pot include **birou** și **ședință**. Dacă apare **până**, răspunsul se îndreaptă spre o oră sau un termen.



Trei momente succesive arată cum fiecare token ales devine context pentru următorul

Răspunsul începe să se formeze odată cu primul token și continuă prin alegeri succesive. Două rulări ale aceleiași cereri pot ajunge

astfel la formulări diferite și ambele pot fi potrivite.

Modelul este mai apropiat de un bucătar decât de o bază de date. Bucătarul folosește ingrediente, tehnici și experiență, apoi gătește preparatul pe loc. O bază de date deschide aceeași înregistrare. Modelul combină tiparele învățate cu ingredientele pe care i le-ai pus în context.

Promptul intră în textul pe care modelul îl continuă

Cererea ta devine începutul răspunsului. **Scrive un email despre întârziere** lasă deschise multe situații. Modelul poate inventa destinatarul, motivul, termenul și tonul.

Dacă îi dai destinatarul, ce s-a întâmplat, promisiunea făcută și noul termen, continuările potrivite se restrâng. Informația relevantă schimbă reprezentările din context și, odată cu ele, probabilitățile tokenilor dintre care modelul îl alege pe următorul.

Modelul poate formula o variantă pe care n-ai scris-o nici tu, nici altcineva exact în forma aceea. Libertatea de a combina tiparele învățate îi permite și să aleagă uneori o direcție nepotrivită. Prima versiune devine material de lucru: o rafinezi când crezi și îi verifici faptele când exactitatea contează.

Răspunsul este construit token cu token: fiecare token ales devine parte din context și schimbă variantele următoare. Informația din prompt restrânge continuările plauzibile, astfel că două rulări pot da formulări diferite. Tratează prima versiune ca pe un draft: verific-o și cere corecții după criterii explicite.

De ce un răspuns fluent poate conține informații false

Cere-i trei studii despre efectul culorii albastre asupra productivității. Modelul a văzut multe bibliografii și știe cum arată: autori, an, titlu, revistă, volum, pagini. Dacă îi lipsește articolul exact, poate umple tiparul cu un titlu plauzibil și un autor care sună potrivit.

Rezultatul arată ca o bibliografie. Propozițiile curg. Uneori lucrarea nu există.



Două răspunsuri la fel de fluente au fundații diferite: unul duce la surse, celălalt la goluri completate

Creierul nostru confundă ușor fluenta cu inteligența

Timp de mii de ani, Homo sapiens a trăit în grupuri în care omul care vorbea fluent, coerent și cu patos era adesea liderul, șamanul sau înțeleptul. Era omul pe care merita să-l ascuți. Creierul nostru și-a format reflexul: vorbește bine, deci știe ce spune.

ChatGPT activează exact acest reflex. Răspunsul curge fără ezitări, apare imediat, este structurat și sună ca vocea unui om care și-a făcut temele. Reflexul nostru vechi ne face să-i atribuim modelului mai multă inteligență decât justifică răspunsul.



Reflexul vechi „vorbește fluent, deci știe” trece prin trei controale: sursă, dovadă și aplicabilitate

Fluența dovedește că modelul se descurcă foarte bine cu limbajul. Exactitatea unei date, existența unei lucrări și corectitudinea unui calcul cer alte probe. Când miza crește, mută atenția de la voce la

proveniență.

Modelele ghicesc când informația lipsește

În lucrarea „Why Language Models Hallucinate” din 2025, Kalai și coautorii descriu un stimulent creat de multe evaluări. Un răspuns greșit primește zero puncte. Un răspuns ghicit poate nimeri și primi punctul. **Nu știi** primește sigur zero în astfel de evaluări, iar această regulă poate încuraja ghicitul.

Astfel de completări apar mai ales la fapte rare: pagina exactă a unui citat, numărul unei hotărâri, titlul unei teze sau numele unei persoane obscure. Forma răspunsului este ușor de învățat. Detaliul exact poate lipsi.

Când informația lipsește, cere-i modelului să pună întrebări, să se abțină sau să caute. Pentru un fapt important, verifică dacă sursa spune ce afirmă răspunsul.

Tonul sigur și incertitudinea sunt lucruri diferite

Asistenții sunt reglați să fie clari și utili. Ezitarea la fiecare propoziție ar face conversația greu de folosit, așa că stilul rămâne adesea hotărât chiar și când modelul are informații puține sau nesigure.

Încrederea modelului, felul în care o exprimă și adevărul afirmației sunt lucruri diferite. Un **probabil** poate însoți o idee corectă. Un **cu siguranță** poate introduce o invenție. Tonul este parte din textul generat.

Cum împinge feedbackul modelul să aprobe utilizatorul

În workshop-uri folosesc uneori aceeași idee de afaceri în două conversații. În prima spun entuziasmat: „Am o idee genială. Vreau să vând «shit on a stick». Ajută-mă s-o dezvolt.” Modelul găsește poziționare, ambalaj și clienți. În a doua spun: „Cineva vrea să vândă «shit on a stick». Mi se pare o prostie. Tu ce zici?” Modelul găsește imediat motivele pentru care ideea va eșua.

Aceasta este tendința de a face pe plac, numită și comportament de **people pleaser**. În post-antrenare, oamenii și modelele de recompensă aleg răspunsurile preferate. Acordul și tonul agreabil pot primi un semnal bun. În analiza „Sycophancy in GPT-4o”, publicată la 29 aprilie 2025, OpenAI a explicat de ce retrăsese o actualizare care ajunsese să se arate prea ușor de acord cu utilizatorii: semnalele de feedback fuseseră ponderate prost.

Poți schimba cadrul întrebării. **Ideea mea este bună?** îi oferă deja direcția. Cere criterii de evaluare, slăbiciuni și condițiile în care ideea eșuează. Răspunsul devine mai util când modelul are o treabă mai precisă decât să-ți confirme entuziasmul.

Fluența, tonul sigur și tendința modelului de a-i da dreptate utilizatorului sunt proprietăți ale textului generat. Adevărul unei afirmații se stabilește prin verificarea faptelor. Când informația exactă lipsește, modelul poate completa un tipar plauzibil; reformulează întrebarea, cere criterii și verifică faptele importante în sursa potrivită.

Contextul și uneltele schimbă rezultatul

Răspunsul folosește parametrii și contextul pregătit de aplicație. În context pot intra conversația, instrucțiuni, exemple, documente și rezultate aduse de unelte. Toate influențează următorul token.

Ce informații din context schimbă răspunsul

Imaginează-ți un coleg nou foarte capabil. Îi spui: „Vezi că balanța aia nu merge.” Colegul nu știe despre ce balanță vorbești, unde se află și ce rezultat așteptai. Inteligența lui nu-i aduce informațiile interne ale companiei.

Cu modelul se întâmplă la fel. Spune-i situația, publicul, rezultatul dorit și informația care schimbă răspunsul. Atașează documentul pe care trebuie să-l folosească. Păstrează informațiile necesare pentru rezultat și elimină materialul fără legătură, duplicat sau depășit.

Dacă documentele conțin date sensibile, consultă echipa IT ori departamentul juridic și obține aprobarea cerută de regulile interne înainte să le încarci.

Uneltele aduc informație și pot executa calcule

Un calculator face aritmetică. O căutare web poate găsi o sursă publicată recent. Codul poate număra literele din **strawberry**. Cu aceste unelte, modelul poate folosi și informații din afara parametrilor.

În proces intervin modelul și aplicația. Modelul cere folosirea unei unelte. Componenta aplicației care coordonează modelul și uneltele, numită orchestrator, execută cererea. Rezultatul revine în context, iar modelul îl folosește pentru continuare.



Modelul propune apelul, orchestratorul folosește unealta, iar rezultatul revine în context pentru verificare

Imaginea arată și locurile în care poți inspecta procesul. Uită-te la întrebarea trimisă de model, apoi la rezultatul unelei. La final, caută afirmația din răspuns care se sprijină pe acel rezultat.

Recalculează un total important, așa cum ai verifica un rezultat primit de la un coleg. Deschide sursa unei afirmații și verifică fragmentul folosit dintr-un document. Pentru un tabel, poți compara totalul cu un calcul independent. Pentru o afirmație, verifici documentul care o susține.

Parametrii codifică tiparele învățate, contextul conține informațiile disponibile acum, iar uneltele pot aduce surse recente sau rezultate calculate. Pentru un rezultat important, urmărește proveniența: ce informație vine din document, ce rezultat a dat unaalță și ce a formulat modelul folosind informațiile și rezultatele primite.

LECȚIA 2 ÎNTR-O PAGINĂ

Cum se construiește un răspuns, token cu token

Un LLM transformă textul în reprezentări numerice și construiește răspunsul prin alegeri succesive. Parametrii păstrează tiparele învățate, contextul aduce informația disponibilă acum, iar uneltele pot furniza date noi. Proveniența te ajută să judeci rezultatul.

1. Reprezentarea numerică

Textul este împărțit în tokeni. ID-ul indică tokenul în vocabular, iar vectorii îl reprezintă în calcule. Fragmentele din jur îi schimbă sensul în context.

2. Ajustarea parametrilor

Antrenarea ajustează parametrii prin multe predicții și corecții. Ei ajung să codifice împreună relații dintre concepte. Modelul nu păstrează sertare cu definiții, ci folosește această structură numerică pentru a continua textul.

3. Construirea răspunsului

Modelul alege următorul token și repetă calculul. Tokenul intră în context și schimbă pasul următor. Promptul influențează alegerile următoare, iar rulări diferite pot da formulări diferite.

4. Fluență și adevăr

Modelul caută o continuare plauzibilă. Când îi lipsește informația exactă, poate completa tiparul cu un detaliu fals, formulat convingător. Tonul sigur nu este o dovadă; verifică afirmațiile importante în sursa potrivită.

5. Proveniența rezultatului

Tiparele vin din parametri; informația actuală, din context și documente; datele noi, din unelte. Urmărește ce informație a intrat, ce pas a produs rezultatul și ce sursă îl susține. Așa separi formularea modelului de informația verificabilă.

Text → tokeni și vectori → calcule → token nou → context actualizat → repetare. Verifică proveniența.

Ce urmează

Acum poți urmări răspunsul de la mesaj până la tokenii generați. Știi cum antrenarea leagă limbajul de spațiul numeric și de ce două răspunsuri la fel de fluente pot conține informații corecte sau invenții.

În lecția 3 trecem de la mecanism la practică. Vei folosi context relevant, pași pe care îi poți inspecta și vei verifica fiecare tip de rezultat așa cum cere situația.

Două experimente până la lecția următoare

1. Pune aceeași întrebare factuală în trei conversații noi. Marchează detaliile care diferă, cere sursele și verifică afirmațiile într-o sursă primară sau independentă.
 2. Prezintă aceeași idee o dată cu entuziasm și o dată critic. Apoi cere o evaluare după criterii fixe și compară cele trei răspunsuri.
-



Viorel Spînu AI Enabler
curs.viorelspinu.com