Værdiskabelse med AI: Nedefra og op

Jakob Mørup Wang

2026

Indhold

Kapitel 1

3.2 LLM

En sprogmodel (LLM) alene er en matematisk funktion.

$y = f(x)$.

Når du skriver en besked til en chatbot, oversætter et program (en /tokenizer/) først dine ord til en serie af tal x inden de sendes videre til sprogmodellen, den matematiske funktion $f(x)$.

Inde i $f(x)$ sker der enormt mange beregninger. Typisk tælles de i milliarderne afhængigt af sprogmodellens størrelse. På en GPU kan beregningerne køre sideløbende i stedet for at vente på hinanden, og derfor kan det gå virkelig hurtigt og skal måles i millisekunder.

Når alle beregningerne er kørt, kommer der i sidste ende ét tal y ud af det. Dette tal oversættes direkte tilbage til et ord med samme ordbog, som blev brugt til at oversætte din besked til tal.

Det er derfor en sprogmodel svarer med ét ord ad gangen. Hvert af disse ord bliver løbende føjet til den oprindelige besked og kørt igennem $f(x)$ igen. Det giver endnu et nyt ord, og sådan kører det i ring, indtil sprogmodellen skriver et stopord, der betyder, at svaret er færdigt, og at programmet ikke skal kalde funktionen $f(x)$ yderligere.

Når sprogmodellen kan generere serier af ord, der fungerer som svar i en samtale, er det fordi, at den modtager hele samtalen i form af et manuskript, som den fortsætter. Det er på den måde, rammen dannes for en simpel chatbot.

I praksis virker det ved, at værdien for hvert enkelt ord beregnes, når sprogmodellens milliarder af interne værdier (ofte kaldet parametre eller vægte) på kryds og tværs ganges eller lægges til de værdier, som dine ord er oversat til.

Sprogmodellens interne værdier, som i øvrigt er helt almindelige decimaltal, er resultatet af dens træning, hvor de er blevet matematisk mikrojusteret trilliarder af gange, når $f(x)$ igen og igen har regnet på bidder af sin meget store mængde træningsdata. Efter træningen (også kaldet fitting; en model fittes eller tilpasses dataen) som kræver enormt store mængder regnekraft, er værdierne dog helt statiske, og der sker ingen justeringer, når den anvendes.

På den måde vil sprogmodellen altså generere et ord y, som i dens træningsdata med tilstrækkelig sandsynlighed følger den forudgående del af samtalen x. Så spørgsmålet er,

hvilket ord der i modellens træningsdata sandsynligvis følger, og hvad der i træningsprocessen er brugt som definition på sandsynlighed?

Medmindre du får værdi af generisk tekst, har sprogmodellen altså brug for et lag mere. Den skal pakkes ind i en applikation, som kan styre den og stille værktøjer til rådighed for den.

#3.3 Applikation

En sprogmodel kan i sig selv ikke handle, og den har ikke adgang til viden udover det, der er indlejret i dens parametre. Den kan først og fremmest generere ord, der bliver til en sammenhængende fortsættelse af en tekst. For at række ud over dette, skal den styres af en applikation.

Når modellen gennem en applikation får adgang til at anvende værktøjer, såsom at slå op i en database eller summere kolonner i et regneark, går den i daglig tale fra at være en *chatbot* til at være en *agent*.

Du kan dog let blive snydt, da en applikation kan være opbygget på en måde, så sprogmodellen ikke har egentlig mulighed for at handle (den har ingen *agens*), men blot modtager ekstra information, som applikationen på forhånd har suppleret din besked til sprogmodellen med.

Denne snyd skyldes, at det er både billigere og lettere bare at udføre funktioner uafhængigt af sprogmodellen og bagefter fortælle den resultatet, end det er at instruere den i hvorfor, hvornår og hvordan den skal udtrykke, at den vil udføre funktionerne. Endnu sværere er det at håndtere, hvis den laver fejl. Det kræver både meget præcise instruktioner og en nøje planlagt grænseflade mellem sprogmodel og applikation.

Til gengæld kan det også resultere i meget mere dybdegående svar og større succes med udførelse af handlinger, fordi sprogmodellen kan forfølge spor fra resultaterne af én søgning til den næste (*multihop-søgning*), eksperimentere med værktøjskald og korrigere dem på baggrund af responsen, eller den kan sammenholde data fra flere kilder.

Sprogmodellens handlinger fungerer på den måde, at den genererer bestemte nøgleord eller særlige tegn, når den vil bruge et værktøj. Applikationen overvåger så modellen for at holde øje med disse nøgleord, ligesom den i forvejen gør for at holde øje med et stopord, der markerer at svaret er færdigt. Genkender applikationen et nøgleord, afbryder den sprogmodellen og udfører den ønskede handling, f.eks. en søgning eller at sende en e-mail.

Herefter kalder applikationen så modellen igen med responsen fra værktøjet som den seneste del af samtalen, fx søgeresultater. Både værktøjskald og respons holdes dog typisk skjult for dig som bruger, men modellen vil ord for ord generere en sandsynlig fortsættelse med afsæt i de sandsynligheder, den har lært af sin træningsdata, hvor denne samtale inklusive værktøjskald og respons er aktuel kontekst.

Denne fortsættelse kan fx være *reasoning*, altså simulering af en indre samtale, hvilket fungerer som en skærpelse af konteksten for dens næste genererede ord. Det kan også være et endeligt svar, der konkluderer på responsen, efterfulgt af et stopord. Eller det kan være endnu et nøgleord, som fortæller applikationen, at der skal udføres endnu en handling.

Denne proces kompliceres af, at en sprogmodel er en statisk funktion. Den har ingen

hukommelse. Den husker ikke, hvad den skrev for to sekunder siden, eller hvad du spurgte om i starten af samtalen. Alt, modellen skal vide for at kunne svare, skal sendes med i den aktuelle kørsel.

Sprogmodellen har samtidig en begrænset arbejdshukommelse. Der er en øvre grænse for, hvor meget tekst den kan modtage ad gangen. Denne øvre grænse kaldes ofte modellens /kontekstvindue/. Selvom brugergrænsefladen viser en lang historik, er det sjældent muligt at sende hele samtalen med til modellen hver gang. Konteksten skal derfor reduceres og prioriteres kraftigt.

For at løse problemet med manglende hukommelse for en chatbot, kan applikationen afkorte konteksten efter simple regler, som fx kun at medtage et bestemt antal af de seneste beskeder, men for en agent er det mere problematisk. Dens kontekst fyldes nemlig hurtigt op af dens indre samtale, værktøjskald og applikationens tilhørende respons herpå.

En løsning for en agent er at lade den generere *artefakter*. Det er tekstbidder, der fungerer som noter eller delkonklusioner, som agenten efterlader til sig selv. Disse artefakter kan sendes med i næste kørsel i stedet for hele den rå datahistorik. Det sparer plads i arbejdshukommelsen. Artefakterne kan også gemmes i en ekstern hukommelse, som agenten kan slå op i med et værktøj, hvis den får brug for at genbesøge en særlig viden, den ellers har udeladt.

Det er for indeværende et åbent forskningsspørgsmål, hvordan denne proces styres stabilt. Det er komplekst at få en sprogmodel til at indhente viden og udføre handlinger over mange trin uden at tabe tråden eller fylde arbejdshukommelsen med støj. Heri ligger en stor begrænsning for værdiskabelsen med sprogmodeller, da det er med til at gøre dem upålidelige, og det stiller meget store krav til, hvordan den data, de har brug for, er tilrettelagt i den bagvedliggende datainfrastruktur.

Kapitel 2

3.4 Datainfrastruktur

Databaser og søgemaskiner er ikke en ny opfindelse. De fleste af os har brugt dem i årevis på den ene eller anden måde. Og kløgtige forskere har løbende opfundet nye metoder og systemer til at tilrettelægge data på måder, så de bliver lettere tilgængelige. Man taler om at indeksere data.

Med sprogmodellernes indtog, har vi mere brug for disse metoder end nogensinde, og vi har brug for yderligere metoder til at kunne fastslå, hvornår den ene metode er bedre end den anden.

For hvor du let kan sætte resultaterne af et opslag eller en søgning i den rette kontekst, der måske implicit omfatter formål, delmål, eksterne bindinger, noget du læste i går, og erfaring fra sidst du slog op i databasen, ja, så kniber det for en sprogmodel. Den har brug for det hele eksplicit, hvis du vil være sikker på, at dens svar og handlinger afspejler det. Ellers genererer den det næste ord på baggrund af den mest sandsynlige kontekst, der er indlejret i den fra dens træning.

Desuden tænker vi slet ikke over mange af de signaler, vi får, når vi søger information. Fx er et link fra én side til en anden et signal om en forbindelse, og teksten omkring linket er en kontekst, der signalerer forbindelsens art. Er det en semantisk forbindelse (*Læs mere her*)? Eller er det en finansiel (*Køb nu*)? Eller er den obligatorisk, men irrelevant (*Cookiepolitik*)?

Bla bla.

Hvordan vi ser det vs. hvordan systemet ser det (uden hjælp)

Figur 2.1: Hvordan vi ser det vs. hvordan systemet ser det (uden hjælp)