https://doi.org/1071066/VARECZA.2025.1.02

Dancs Sándor

MESTERSÉGES INTELLIGENCIA ALAPÚ ONLINE PROGRAMOZÁSI FELADATBANK ÉS KIÉRTÉKELŐ RENDSZER LÉTREHOZÁSA

DEVELOPING AN ARTIFICIAL INTELLIGENCE-BASED ONLINE PROGRAMMING EXERCISE BANK AND EVALUATION SYSTEM

Összefoglaló

Az előadás a konferencia első napján hangzott el. Az igen érdekes matematikai előadásokat követően, először meghatároztam az MI típusait olyan megközelítésben, hogy el tudjam helyezni az általam használt szoftvereket, aztán kitértem az MI felhasználási területeire is az oktatásban. Majd a problémamegoldás, programozás oktatás chatbotokkal való támogatását vizsgáltam. Bemutattam a chatbotok hallgatóim általi használatával kapcsolatos megállapításokat, tapasztalatokat a hallgatói és az oktatói oldalról is, illetve felsoroltam, hogy mire használják a programozás tanulás során. Mindezekből következett az a megállapítás, hogy az oktatási intézményeknek az MI használatához alkalmazkodniuk kell! Feltettem a kérdést, hogy a programozás oktatás milyen szegmenseit érdemes kiszervezni az MI eszközökbe. A hallgatók általában először kód generálására használják a chatbotokat, egy adott feladat megoldásához, a feladatokat kaphatják az órákon, választhatják feladatbankokból, generáltathatják vagy oldhatnak meg saját problémákat is. Bemutattam feladatbankokat és kiértékelő rendszereket, beszéltem az előnyeikről és a hátrányaikról. Megfogalmaztam, hogy cél, egy MI alapú online programozási feladatbank és kiértékelő rendszer létrehozása! Végül, meghatároztam az új rendszer funkcióit, tulajdonságait, kitértem arra, hogy miért fontos és milyen a jó prompt, illetve a generált tartalmak felhasználásakor, megosztásakor figyelembe veendő etikai normákra, szabályozási iránymutatásokra.

Kulcsszavak: Mesterséges Intelligencia, programozás oktatás, feladatbank, kiértékelés, prompt, generálás, etika

Abstract

The presentation was delivered on the first day of the conference. Following several highly engaging mathematical talks, I began by defining the types of Artificial Intelligence in a way that allowed me to position and contextualize the software tools I use. I then discussed the applications of AI in education. After this, I examined the use of chatbots to support problem solving and programming instruction. I presented observations and experiences related to how my students use chatbots - both from the students' and instructors' perspectives - and listed the ways in which they rely on it while learning to program. Based on these findings, I concluded that educational institutions must adapt to the use of AI. I raised the question of which segments of programming education are worth outsourcing to AI tools. Students typically start by using chatbots to generate code for solving a given task. These tasks may be provided during class, selected from exercise banks, generated automatically, or derived from students' own problems. I showed several exercise banks and automated evaluation systems, and discussed their advantages and disadvantages. I then outlined the goal of developing an AI-based online programming exercise bank and evaluation system. Finally, I defined the key functions and properties of such a system, emphasized the importance of crafting effective prompts, and addressed the ethical norms and regulatory guidelines that must be considered when using and sharing AI-generated content.

Keywords: Artificial Intelligence, programming education, exercise bank, evaluation, prompt, generation, ethics

1. Bevezetés

Kétszáz éves idén a Magyar Tudományos Akadémia, az országos tudományünnepi programsorozathoz ismét csatlakozott a Nyíregyházi Egyetem. A jubileumi ünnep alkalmával

az egyetem valamennyi intézete saját tudományos programsorozattal készült. A sorozat központi eleme, "A Tudomány és Művészet Ünnepe a Nyíregyházi Egyetemen" esemény volt, melynek mottója: "200 év a tudás és a társadalom szolgálatában". Az "Ember és mesterséges intelligencia szimbiózisa" című ünnepi plenáris előadást Prof. Dr. Szűts Zoltán, az Eszterházy Károly Katolikus Egyetem dékánja tartotta. A programot Prof. Dr. habil. Simon László, a Nyíregyházi Egyetem Tudományos Tanácsának elnöke vezette. Idézet a professzortól: "A mesterséges intelligenciát ma már a felsőoktatásban sem lehet megkerülni, hiszen segíti a kutatást, az oktatást, és új kérdéseket vet fel a tudományos gondolkodásban." A Magyar Tudomány Ünnepe programsorozat részeként, a Matematika és Informatika Intézet által rendezett, "Dr. Varecza Árpád Tudományos Emlékülés és Konferencia" ünnepi programján megtartott előadásom jól illeszkedett ehhez a gondolatmenethez, illetve a plenáris előadáshoz is.

Az előadás (prezentáció, fotók) itt érhető el: https://nye.dancs.org/publikaciok/

A prezentáció első és utolsó diáira az előadás előtt tíz perccel generáltam két képet. Az OpenAI DALL-E szövegből képbe AI képgeneráló rendszert használtam. (Internet 1)

Prompt: Szia, egy konferencián fogok előadást tartani. Az előadásom címe: Mesterséges Intelligencia alapú online programozási feladatbank és kiértékelő rendszer létrehozása. Kérlek, hogy készíts két képet, az első az előadás előtti motivációról, figyelemfelkeltésről szóljon, a második az előadás zárásaként funkcionáljon!

Aztán kérdést kaptam a stílusra és feliratokra vonatkozóan. Választhattam a felajánlott modern, technológiai, illetve letisztult, tudományos-akadémiai stílusok közül.

Prompt: Inkább modern, technológiai (pl. kék neon, digitális hálózat, AI-esztétika) stílust szeretnék, feliratokra nincs szükségem.

Az 1. ábrán látható az előadás első diája, a generált, motivációs, figyelemfelkeltő és AI-esztétikájú nyitóképpel.

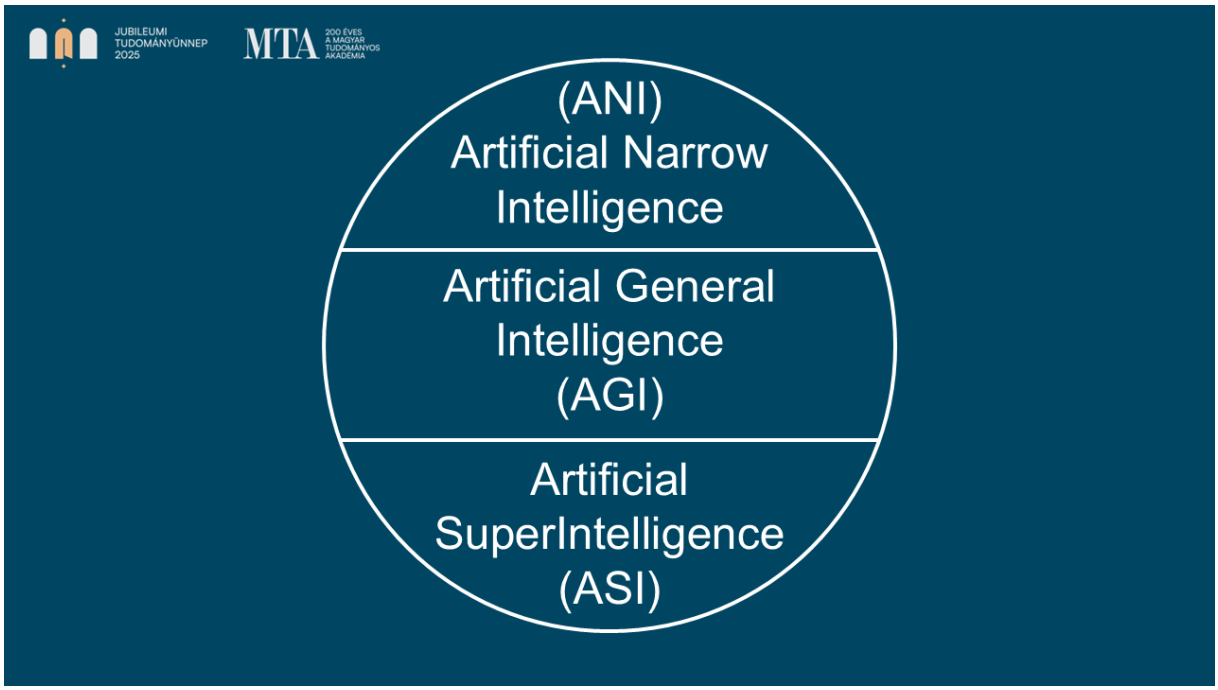
1. ábra: Előadás első diája

A 2. ábrán látható az előadás utolsó diája, a generált, inspiráló, lezáró és jövőbe mutató hangulatú záróképpel.

2. ábra: Előadás utolsó diája

Az MI az élet szinte minden területén jelen van, nem maradhat figyelmen kívül az oktatási intézményekben sem! Ezért fontos kérdés, hogy az oktatás szereplői az MI alapú szoftvereket hogyan alkalmazzák az oktatásban, illetve beépítik-e a tanítási, illetve tanulási folyamataikba?

A 3. ábrán (előadás: 5. dia) jól láthatóan meghatároztam először az AI (Artificial Intelligence) típusait olyan megközelítésben, hogy el tudjam helyezni az általam használt szoftvereket.

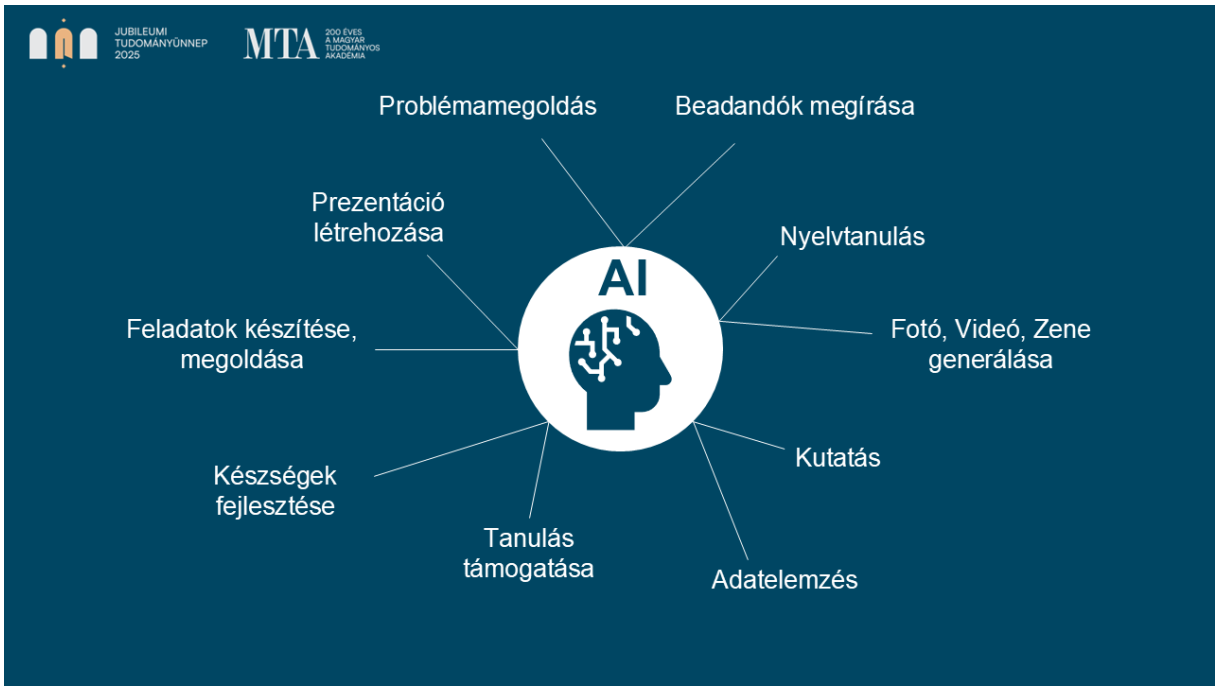


3. ábra: AI típusai

A résztvevőket megismertettem, a nagy nyelvi modelleken (LLMs, Large Language Models) alapuló, AI szoftverek alkalmazási lehetőségeivel. Ilyenek a generatív előképzett transzformátorok, (GPTs, Generative Pre-trained Transformers) a generatív AI (GAI, Generative AI) keretrendszerei, pl. a chatbot (csevegőrobot) ChatGPT. (Chan - Colloton, 2024; Russell - Norvig, 2023) Aztán előre vetíttettem a kérdést, hogy használatosak-e ezek a szoftverek a diákok, hallgatók által a programozás tanulásuk támogatására?

Feltételeztem, hogy az oktatás szereplői egyre inkább beépítik a tanítási, illetve tanulási folyamataikba az AI alapú szoftverek használatát, hiszen képesek pl.: szövegeket, kódokat, képeket, hangokat és videókat, a generatív Mesterséges Intelligencia modellek segítségével generálni.

Kitértem az AI felhasználási területeire is az oktatásban (4. ábra, előadás: 8. dia).



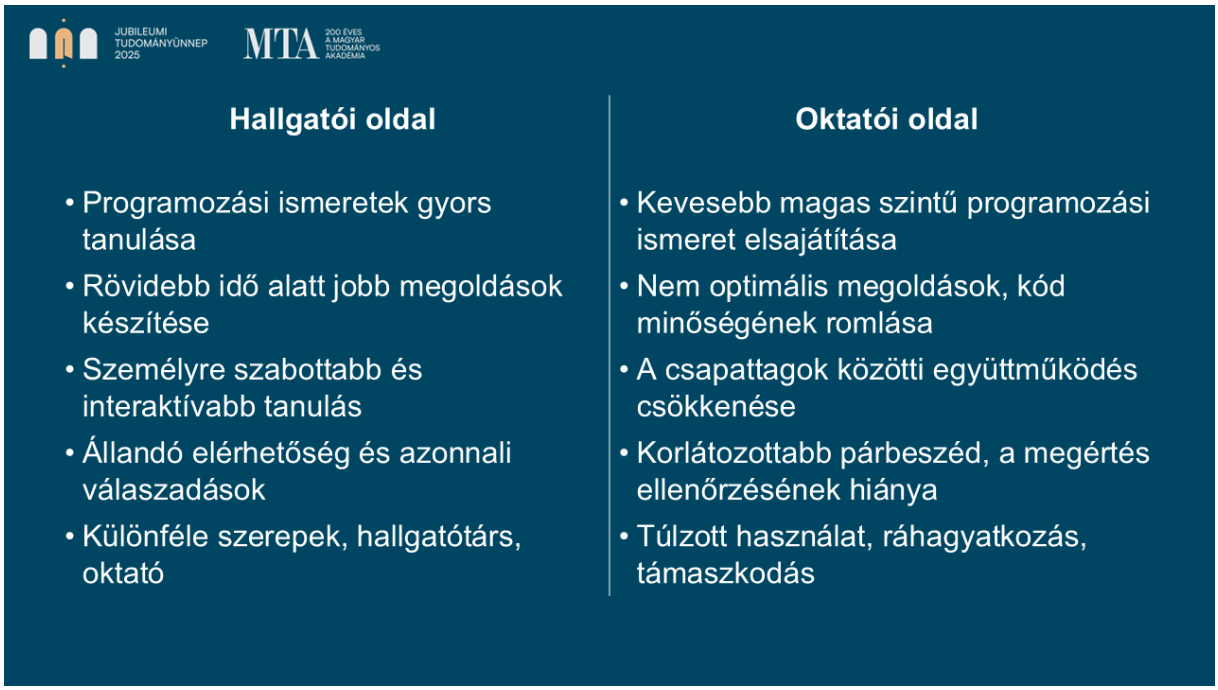
4. ábra: AI felhasználási területei az oktatásban

Majd a problémamegoldás, programozás oktatás chatbotokkal való támogatását vizsgáltam. Bemutattam, a chatbotok hallgatóim általi használatával kapcsolatos megállapításokat, tapasztalatokat a hallgatói és az oktatói oldalról is (5. és 6. ábra, előadás: 11. és 12. dia), illetve felsoroltam, hogy mire használják a programozás tanulás során.

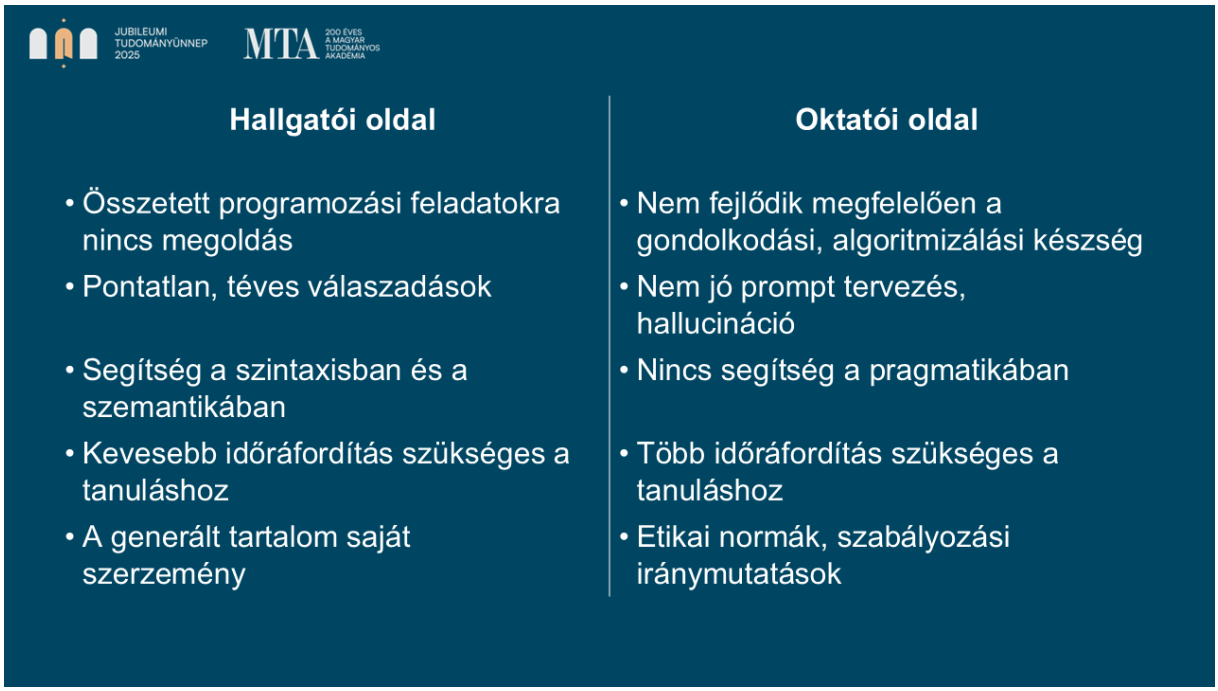
A hallgatóim a chatbotokat használták:

- kód generálására,
- kód magyarázatára,
- kód hibaellenőrzésére és hibakeresésére,
- kód optimalizálására,
- programozási ismeretek elsajátítására,
- és programozási feladatok generálására.

A lista csökkenő gyakoriság szerint rendezett. A használattal kapcsolatos megállapításokat, tapasztalatokat érdemes volt összevetni a hallgatói és az oktatói oldalról nézve!



5. ábra: A chatbotok hallgatóim általi használatával kapcsolatos megállapítások, tapasztalatok a hallgatói és az oktatói oldalról



6. ábra: A chatbotok hallgatóim általi használatával kapcsolatos megállapítások, tapasztalatok a hallgatói és az oktatói oldalról

Mindezekből következett az a megállapítás, hogy az oktatási intézményeknek az MI használatához alkalmazkodniuk kell!

Feltettem a kérdést, hogy a programozás oktatás milyen szegmenseit érdemes kiszervezni az MI eszközökbe!

A hallgatók általában először kód generálására használják a chatbotokat, egy adott feladat megoldásához, a feladatokat kaphatják az órákon, választhatják feladatbankokból, generáltathatják vagy oldhatnak meg saját problémákat is.

Bemutattam két online programozási feladatbankot és kiértékelő rendszert, illetve beszéltem az előnyeikről és a hátrányaikról:

ELTE-IK MESTER

DE-IK Progcont

Megfogalmaztam, hogy cél, egy MI alapú online programozási feladatbank és kiértékelő rendszer létrehozása!

Végül, meghatároztam az új rendszer funkcióit, tulajdonságait, - kitértem arra, hogy miért fontos és milyen a jó prompt, illetve a generált tartalmak felhasználásakor, megosztásakor figyelembe veendő etikai normákra, szabályozási iránymutatásokra:

- Több nyelv támogatása
- Nehézségi szintek választhatósága
- Programozási feladatok hozzáadása
- Programozási feladatok generálása
- Generált feladatok eltárolása
- Megoldások ellenőrzése
- Felhasználói jelzések
- Számonkérések
- Versenyek

2. Több nyelv támogatása

Javasolt az alábbi nyelvek támogatása, természetesen a lista bővíthető, változtatható:

- C
- C++
- C#
- Java
- Pascal
- Python

3. Nehézségi szintek választhatósága

Érdemes megfontolni a nehézségi szintek választhatóságát, pl.: kezdő, középhaladó, haladó és extrém / verseny. Bár a feladatok különböző szintekbe sorolása szubjektív, viszont így lehetőségük van a rendszer felhasználóinak arra, hogy az aktuális tudásuknak, tapasztalatuknak megfelelő feladatok megoldásával fokozatosan fejlődjenek a tanulási folyamatuk során.

4. Programozási feladatok hozzáadása

Fontos, hogy az oktatók tudjanak a rendszerhez feladatokat hozzáadni és azokat különböző nehézségi szintekbe sorolni.

Az oktatók használhatják a rendszert számonkérésekre is, pl. nagy létszámú kurzusokon, ahol a kézi ellenőrzés időigényes és ami könnyen lehet szubjektív. Fontos a visszacsatolás, az

oktatók nyomon követhetik a hallgatók fejlődését, ennek megfelelően változtathatnak a tananyagokon, módszereken.

Egy programozási feladatot pontosan leíró specifikáció a következőket kell, hogy tartalmazza:

- Feladat neve
- Feladat szövege
- Bemenet megfogalmazása
- Kimenet megfogalmazása
- Példa (bemenet, kimenet) - első teszteset
- Időkorlát
- Memóriakorlát
- Tesztesetek

A feladat neve legyen kifejező, hiszen sokan ez alapján választanak feladatot, ha érdekesnek találják vagy kíváncsiak, mert nincs idejük több feladat elolvasására. A feladat szövege legyen világos, egyértelmű, jól értelmezhető. A bemenet megfogalmazása pontosan adja meg, hogy milyen adatok találhatók a standard bemeneten, milyen tartományból vehetnek fel értéket, illetve az adatok maximális mennyiségét, ha nem akarunk dinamikus memóriakezelést. A kimenet megfogalmazása pontosan adja meg, hogy milyen adatokat milyen elválasztó jelekkel kell kiírni a standard kimenetre. Az egyszerű példa - az első teszteset - tartalmazzon rövid bemenetet és a hozzá tartozó helyes, rövid kimenetet. Az erőforrás korlátok (processzor idő és memória) szükségesek a megoldások kiértékelésekor, gondolva a nagy erőforrásigényű megoldásokra, esetlegesen rossz indulatú kódra, végtelen ciklusra, ezekben az esetekben ugyanis a kiértékelést idő- vagy memóriatúllépési hibával le kell állítani. A korlátok átlépése visszajelzés a felhasználónak, hogy hibás vagy - tipikusan a nagyobb, összetettebb tesztesetek esetében - nem optimális az algoritmusa. A konkrét tesztesetek meghatározása - a specifikáció elkészítése mellett - nagyon időigényes feladat és gyakran hiányoznak a teljes körű ellenőrzéshez még esetek. A feladathoz generálhatunk teszteseteket is. A felhasználók a feladatot és az abban lévő példán kívül, még az utolsó (összetettebb) tesztesetet tölthetik le, az oktatók pedig mindegyiket.

Hogyan fogjuk feltölteni feladatokkal az online programozási feladatbankot?

5. Programozási feladatok generálása

A felhasználók programozási feladatokat generálhatnak pl. a GPT-5.1 modell (Internet 2) használatával, amihez az új rendszerünkből való hozzáférést az OpenAI API (Application Programming Interface, Alkalmazásprogramozási felület) teszi lehetővé. (Internet 3)

A modellenként eltérő árazást itt lehet megtekinteni: https://platform.openai.com/docs/pricing

Az árak, szöveges tokenek (lehet egy egész szó, egy szó egy része, írásjel, szóköz vagy akár egyetlen karakter is) és GPT-5.1 modell esetében (az árak 1 millió tokenre vonatkoznak):

Bemeneti tokenek: $1.25
Gyorsítótárazott bemeneti tokenek: $0.125
Kimeneti tokenek: $10.00

A legalább 1024 tokent tartalmazó promptok esetében, a tokenek és a hozzájuk tartozó belső számítási eredmények automatikusan a gyorsítótárba kerülnek. A következő lekérdezésekben, az azonos vagy hasonló részek esetében nem kell újra elvégezni a számításokat, hanem

betöltődik a már kiszámolt és eltárolt belső reprezentáció, így gyorsabb lesz a válasz és olcsóbb a használat. Szóval, csak a bemeneti tokenek (prompt) és a számítási eredmény tárolódik el, a kimeneti tokenek (válasz) mindig újra generálódnak. A gyorsítótárazott promptok törlése is automatikus, ha egy ideig (jellemzően 5 - 10 percig, de ha a rendszer nem terhelt, akkor akár egy óráig) nem érkezik azonos vagy hasonló kérés, akkor kerül rá sor. Lehet kérni 24 órás gyorsítótárazást (prompt_cache_retention: "24h"), ha az adott modell támogatja, de csak akkor garantált, hogy a prompt ennyi ideig tárolva lesz, amikor a körülmények éppen engedik, pl. a rendszer túlterhelése, nagy memóriaigénye esetén megtörténik a törlés. (Internet 4) A kimeneti tokenek drágábbak, hiszen ez a modell által válaszként adott tokenek költsége.

Hogyan tudjuk csökkenteni a költségeket?

Mivel mindig ugyanaz a promptunk, jó megoldás lenne, ha eltárolnánk a prompt belső reprezentációját és ezt küldenénk a modellnek, hogy a számításokat már ne kelljen elvégeznie, de ezt számos okból nem tehetjük meg, eleve az API vissza sem adja. Néhány dolgot azért megtehetünk, pl.: Tokenizáljuk a promptot, - mivel ez is pénzbe kerül, - tároljuk el a token listát és tokeneket küldjünk a modellnek! Próbáljuk meg lerövidíteni, átfogalmazni, újra tervezni a promptot, úgy, hogy ugyanolyan hatékony legyen! A promptban kérjük a választ (feladat szövege, bemenet, kimenet) röviden, de érthetően megfogalmazni, pl.: tippek, egyéb megjegyzések, magyarázatok, formázás nélkül, jelezve, hogy programban fogjuk feldolgozni.

A promptban a választ kérjük egy JSON (JavaScript Object Notation, JavaScript objektumjelölés) objektumba!

Így nem kell a szöveges válaszban megkeresnünk a feladat nevét, szövegét, a be- és kimeneteket, a teszteseteket és megoldásokat, mert ezeket jól elkülönítve kapjuk.

A hangsúly a prompt tervezésen van!

Miért fontos és milyen a jó prompt?

Más eszközünk nincs, máshogy nem tudunk kommunikálni a modellel. Fogalmazzunk meg egyértelmű, világos, tömör kéréseket, adjuk meg a kontextust, határozzuk meg a célt, a modell szerepét, feladatát, azt, hogy pontosan mit várunk el tőle, a válasz formátumát, struktúráját! Folyamatosan hangoljuk, finomítsuk, teszteljük, optimalizáljuk a promptot, amíg az elvárt eredményt el nem érjük! (Internet 5)

Tesztesetek és megoldások generálása

A korlátok és tulajdonságok figyelembevételével a szükséges teszteseteket generáljuk a modell segítségével, a feladat teljeskörű ellenőrzéséhez! Hívjuk fel a figyelmet a szélsőséges esetekre is! Generáljunk minden támogatott nyelven megoldásokat a generált feladat ellenőrzéséhez!

JSON objektum kulcsok (minden érték sztring):

- feladat_neve
- feladat_szovege
- bemenet
- kimenet
- tesztesetek: tömb (bemenet és kimenet kulcsokkal)
- megoldasok: tömb (c, c++, c#, java, pascal és python kulcsokkal)

Az eddig leírtak alapján és módon készítsük el a végleges promptot!

Generált feladat ellenőrzése

A generált forráskódokat fordítsuk le (amiket szükséges), majd a programokat futtassuk biztonságosan, idő- és memóriakorláttal külön konténerekben párhuzamosan és a generált tesztesetekkel ellenőrizzük le őket! Megfontolhatjuk, hogy ellenőrizzük plusz egy másik modellel is. Mérjük meg a futási időket és a memóriahasználatokat, majd vegyük a legnagyobb értékeket és enyhén becsüljük felül őket a generált feladat specifikációjában való idő- és memóriakorlátok meghatározásához!

Hiba (fordítás, teszt) esetén a feladatot töröljük és generáljunk újat!

Naplózzunk, készítsünk statisztikát arról, hogy hány feladatot dobtunk el és milyen hibák miatt, ez hasznos a prompt újra tervezéséhez!

Konténeralapú végrehajtás és biztonság

A programok külön konténerben való biztonságos futtatásához idő- és memóriakorlátra van szükség. Fontos, hogy a konténereknek ne legyen hálózati hozzáférésük és futtatás után automatikusan törlődjenek.

Cél, a felhasználók egymástól és a rendszertől való teljes elkülönítése.

6. Generált feladatok eltárolása

A generált feladatot felhasználói aktivitás után tároljuk el adatbázisban (az MI által generált feladatok közé), akkor, ha a felhasználó tudta értelmezni és küldött be megoldást rá, az ellenőrzés után kapott eredményt, nincs negatív visszajelzése, azaz elfogadhatónak tartja!

A generált tartalmak felhasználásakor, megosztásakor térjünk ki a figyelembe veendő etikai normákra, szabályozási iránymutatásokra.

Naplózzunk, készítsünk statisztikát arról, hogy hány feladatot dobtunk el és milyen okokból, ez hasznos a prompt újra tervezéséhez!

7. Megoldások ellenőrzése (skálázhatóság)

A felhasználók által beküldött kéréseket (forráskódokat) állítsuk sorba és fordítsuk le (amiket szükséges), majd a programokat futtassuk biztonságosan, a feladat specifikációjában megadott idő- és memóriakorláttal, külön konténerekben párhuzamosan és a generált tesztesetekkel ellenőrizzük le őket! Mérjük meg a futási időt, a memóriahasználatot és a megoldási idővel (a feladat kiválasztásától a megoldásáig eltelt idő), illetve a teszteredményekkel együtt tároljuk el ezeket az adatokat adatbázisban!

A felhasználóknak az adatbázisból küldjünk visszajelzést a megoldási időkről, az esetleges fordítási / futtatási hibákról, az idő- és memóriahasználatról, a helyes és helytelen kimenetekről!

Állítsuk be a maximális konténerszámot!

8. Felhasználói jelzések

Negatív visszajelzést küldhet a felhasználó a feladatról, ha a megoldása során nem érthető, értelmezhető a feladat szövege, a bemenet, a kimenet, illetve hibás tesztesetek (letölthető) esetén, továbbá, ha nem kapott meg valamilyen eredményt, ekkor ne tároljuk el adatbázisban!

A felhasználó visszajelezhet akkor is, ha a feladat a rendszer egy másik feladatával megegyezik vagy arra nagyon hasonlít, ekkor az ismétlődéseket szüntessük meg manuálisan!

Naplózzunk, készítsünk statisztikát arról, hogy hány feladatot dobtunk el és milyen okokból, ez hasznos a prompt újra tervezéséhez!

9. Számonkérések

ZH-k, vizsgák lebonyolítása. Azonnali ellenőrzés, visszajelzés, eredmény. Időmegtakarítás a javítások során (nagy létszámú kurzusok). Diákok, hallgatók fejlődésének nyomon követése.

10. Versenyek

Tehetséggondozás, versenyekre való felkészítés. Különböző (középiskolai, egyetemi, házi, regionális, országos, nemzetközi) versenyek lebonyolítása. Diákok, hallgatók motiválása.

Felhasznált szakirodalom

- Cecilia Ka Yuk Chan - Tom Colloton (2024): Generative AI in Higher Education - The ChatGPT Effect (First Edition), Routledge, London, ISBN: 9781032599045
- Ian Goodfellow - Aaron Courville - Yoshua Bengio (2016): Deep Learning (https://www.deeplearningbook.org/), The MIT Press, Cambridge, Massachusetts, ISBN: 9780262035613
- Stuart Russell - Peter Norvig (2023): Mesterséges intelligencia - Modern megközelítésben I. kötet (Negyedik Kiadás), Taramix Kiadó, Budapest, ISBN: 9786155186769
- Stuart Russell - Peter Norvig (2023): Mesterséges intelligencia - Modern megközelítésben II. kötet (Negyedik Kiadás), Taramix Kiadó, Budapest, ISBN: 9786155186943
- Vinod Chandra S.S. - Anand Hareendran S. (2014): Artificial Intelligence and Machine Learning (First Edition), PHI Learning, Delhi, ISBN: 9788120349346
- Vinod Chandra S.S. - Anand Hareendran S. (2020): Artificial Intelligence - Principles and Applications (Second Edition), PHI Learning, Delhi, ISBN: 9789389347838

További források

- OpenAI Platform: Images and vision URL: https://platform.openai.com/docs/guides/images-vision (2025.11.12.)
- OpenAI Platform: GPT-5.1 URL: https://platform.openai.com/docs/models/gpt-5.1 (2025.11.12.)
- OpenAI Platform: API Platform URL: https://openai.com/api/ (2025.10.08.)
- OpenAI Platform: Prompt caching URL: https://platform.openai.com/docs/guides/prompt-caching (2025.10.06.)
- OpenAI Platform: Text generation URL: https://platform.openai.com/docs/guides/text-generation (2025.10.22.)

SZERZŐI ADATOK

Dancs Sándor mesteroktató
Nyíregyházi Egyetem
Matematika és Informatika Intézet
dancs.sandor@nye.hu