Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Mesterséges Intelligencia Tanszék

Nyílt nyelvi modellek finomhangolása emberi preferenciák alapján

Szakdolgozat

Készítette
Kiss Blanka Zselyke

Konzulens
dr. Gyires-Tóth Bálint
dr. Váradi Tamás

2024. december 5.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Kiss Blanka Zselyke*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2024. december 5.

Kiss Blanka Zselyke
hallgató

Kivonat

Az emberi preferenciákon alapuló finomhangolási algoritmusok új megközelítést kínálnak a modellek teljesítményének javítására, mivel a hagyományos módszerekkel szemben lehetővé teszik az emberi visszajelzések beépítését a modellek optimalizálásába. Ehhez a modellek válaszainak emberi értékelését használják fel. Így a nyelvi modellek képesek pontosabban megfelelni a specifikus követelményeknek, és a válaszaikat az emberi preferenciákhoz igazítani.

A területen a két legelterjedtebb algoritmus a Reinforcement Learning from Human Feedback (RLHF) és a Direct Preference Optimization (DPO). A fő különbség közöttük, hogy az RLHF megerősítéses tanulást használva optimalizálja a modellt az emberi visszajelzések alapján, míg a DPO közvetlenül használja fel az emberi preferenciákat a nyelvi modell optimalizálására. A dolgozatomban részletesen bemutatom ezeket az algoritmusokat.

A DPO előnye az RLHF-fel szemben, hogy egyszerűbb az implementációja, a számításigénye kisebb, és a teljesítménye így is legalább olyan jó, mint az RLHF-nek. Ezért a dolgozatomban DPO alapú finomhangolást valósítottam meg. Az angol nyelvű finomhangolást egy kisebb modellen, a Facebook OPT 350M modellen végeztem el, a magyar nyelvű finomhangoláshoz pedig egy nagyobb modellből, a Llama 3.1 8B Instruct modellből indultam ki. Az angol nyelvű finomhangoláshoz nyílt pénzügyi és matematikai adatbázisokat használtam, a magyar nyelvű adatokat pedig egy angol nyelvű adatbázis fordításával állítottam elő. A folyamatot automatizáltam az OpenAI API segítségével, és a fordításhoz a GPT-4 modellt használtam.

A finomhangolás során az erőforrások optimalizálása érdekében egy Parameter-Efficient Fine-Tuning (PEFT) algoritmust használtam, a Low-Rank Adaptation-t (LoRA), amely jelentősen csökkentette a memóriaigényt. Az angol modell kiértékelését korlátozott számú teszt bemeneten végeztem el, míg a magyar modell kiértékeléséhez összeállítottam egy 100 kérdésből álló adathalmazt. A kérdések részben a Google BIG-bench kérdéssorából, részben pedig saját kritériumok alapján kerültek kiválasztásra.

A dolgozatom fókuszában az emberi preferencia alapú finomhangolási módszerek elemzése, a DPO finomhangolás angol és magyar nyelvű implementációjának elkészítése, valamint a hatékonyságának tesztelése áll. A fő eredményem, hogy a magyar nyelvű DPO finomhangolással az esetek 37%-ában sikerült javítanom a modell teljesítményén.

Abstract

Human preference-based fine-tuning algorithms offer a novel approach to improving model performance, as they enable the integration of human feedback into model optimization, unlike traditional methods. These algorithms rely on the human evaluation of model outputs. This enables language models to better meet specific requirements and tailor their responses to human preferences.

The two most widely used algorithms in this field are Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO). The primary difference between these approaches lies in their methodologies: RLHF uses reinforcement learning to optimize the model based on human feedback, whereas DPO directly leverages human preferences to refine the language model. In my thesis, I provide a detailed overview of these algorithms.

DPO has advantages over RLHF, including simpler implementation, lower computational requirements, and comparable performance. Consequently, my thesis focuses on implementing DPO-based fine-tuning. For fine-tuning in English, I used a smaller model, Facebook's OPT 350M, while for Hungarian fine-tuning, I employed a larger model, the Llama 3.1 8B Instruct model. The English fine-tuning process utilized open financial and mathematical datasets, while the Hungarian dataset was created by translating an English dataset. I automated the process using the OpenAI API, employing the GPT-4 model for translation.

To optimize resources during fine-tuning, I used a Parameter-Efficient Fine-Tuning (PEFT) algorithm, Low-Rank Adaptation (LoRA), which significantly reduced memory requirements. The English model was evaluated using a limited number of test inputs, whereas for the Hungarian model, I compiled a dataset of 100 questions. These questions were selected partly from Google's BIG-bench question set and partly based on custom criteria.

The focus of my thesis is the analysis of human preference-based fine-tuning methods, the implementation of DPO fine-tuning for both English and Hungarian, and testing its efficiency. My main result is that Hungarian DPO fine-tuning improved model performance in 37% of the cases.

1. fejezet

Bevezetés

Az elmúlt évtizedekben a természetes nyelvfeldolgozás (Natural Language Processing, NLP) területe jelentős fejlődésen ment keresztül, amelyet a mesterséges intelligencia (Artificial Intelligence, AI), azon belül is a mélytanulás (Deep Learning) egyre kifinomultabb módszereinek megjelenése tett lehetővé. A természetes nyelvfeldolgozás célja olyan modellek fejlesztése, amelyek képesek a természetes nyelvű szövegekben levő tudás feldolgozására, és ezáltal különböző feladatok elvégzésére, mint például a szövegek generálása, tartalom összefoglalása vagy akár a gépi fordítás. A terület fejlődésében fontos szerepet játszott a nagy nyelvi modellek (Large Language Model, LLM) megjelenése, amelyek több milliárd paraméterrel rendelkezhetnek, és hatalmas mennyiségű szöveges adat feldolgozására képesek.

A transzformer architektúra [24] 2017-es megjelenése nagy áttörést hozott a természetes nyelvfeldolgozás területén, mivel teljesen átalakította a nyelvi modellek működését. A transzformer alapú modellek figyelemmechanizmusa (attention mechanism) lehetővé teszi, hogy minden egyes szót annak teljes szövegkörnyezete alapján értelmezzenek, így a szavak jelentését nem elszigetelten, hanem a teljes mondat vagy szöveg kontextusának figyelembevételével számítják ki. Ez hatalmas előrelépést jelentett a korábbi szekvenciális modellekkel szemben, amelyek csak egy adott sorrendben tudták feldolgozni az információt, így nehezen kezelték a szavak közötti függőségeket. Emellett a transzformerek rendkívül jól skálázhatóak, ami lehetővé tette a modellek paramétereinek exponenciális növekedését, akár több milliárdos nagyságrendig. Ennek köszönhetően a mai nyelvi modellek nagy része a transzformer architektúrára épül.

Bár ez a technológia a szakemberek számára régóta ismert volt, a nagyközönség számára csak 2022-ben, az OpenAI ChatGPT modelljének megjelenésével vált egyértelművé a benne rejlő potenciál. Az OpenAI egy egyszerűen kezelhető felhasználói felületet készített a modellhez, amely az átlagember számára is lehetővé tette, hogy megtapasztalja a valós idejű ember és gép közötti kommunikációt. Ezt követően számos további chatbotot adtak ki, így a legfejlettebb nyelvi modellek is széleskörben elérhetővé váltak. Ezek a modellek hatalmas segítséget nyújtanak a mindennapi élet feladataiban, mivel folyamatosan rendelkezésre állnak, és mindenki számára ingyenesen elérhetőek. Legyen szó kreatívabb feladatokról, mint egy vers megírása, vagy egy kötöttebb matematikai problémáról, a nyelvi modellek használata jelentősen gyorsíthatja a felhasználók munkáját. Ezt jól érzékelteti, hogy azok a programozók, akik GitHub Copilot-ot használnak kódírás során, több, mint másfélszer olyan gyorsan tudják elvégezni a feladatukat, mint az önállóan dolgozó kollégáik [17].

A nagy nyelvi modellek széleskörű alkalmazhatósága számos előnnyel jár, de egyben jelentős kihívásokat is okoz. A különböző feladatok esetén felmerülő eltérő elvárások nagymértékben megnehezítik a modell fejlesztését és kiértékelését. Például egy programozási

feladatnál elvárható, hogy a modell által generált kód futtatható legyen, míg egy történet megírásánál a kreativitás előnyös. A hagyományos modelleknél a teljesítmény értékelése legtöbbször egy veszteségfüggvény alapján történik, ami azt méri, hogy a modell kimenete milyen mértékben tér el az elvárt eredménytől. Azonban a látszólag egymásnak ellentmondó elvárások miatt nehéz egyetlen veszteségfüggvényben összefoglalni az összes olyan szempontot, amelyek egy adott feladatnál fontosak.

Ennek a problémának a megoldására olyan új megközelítést alkalmazó módszerek születtek, amelyek figyelembe veszik az emberi preferenciákat. Ezek közül a legismertebbek a Reinforcement Learning from Human Feedback (RLHF) és a Direct Preference Optimization (DPO). Ezeknél az algoritmusoknál az emberek közvetlenül értékelik a modell által generált kimeneteket, és ezeket az értékeléseket használják fel a modell fejlesztésére. Ezáltal a nyelvi modellek képesek lesznek pontosabban megfelelni a specifikus követelményeknek, és a válaszaikat az emberi preferenciákhoz igazítani. A két módszer között a fő különbség, hogy az RLHF megerősítéses tanulást használva optimalizálja a modellt az emberi visszajelzések alapján, míg a DPO közvetlenül használja fel az emberi preferenciákat a nyelvi modell optimalizálására.

A dolgozatom második fejezete tartalmazza az emberi preferenciákon alapuló finomhangolási algoritmusok áttekintését. Bevezetek néhány természetes nyelvfeldolgozással kapcsolatos alapfogalmat, amelyek szükségesek a módszerek megértéséhez, valamint részletesen bemutatom az RLHF és DPO működését, és összehasonlítom őket. Itt feltételezem, hogy az olvasó tisztában van a neurális hálók alapfogalmaival. A harmadik fejezetben röviden meghatározom az implementációval kapcsolatos céljaimat. A negyedik fejezetben bemutatom az implementációhoz használt eszközöket és algoritmusokat. Ezután az ötödik fejezetben ismertetem az angol nyelvű DPO megvalósításának lépéseit, majd a hatodik fejezetben áttérek a magyar nyelvű DPO implementáció részletezésére, ahol kitérek a kiértékelés módszertanára, és elemzem az eredményeket. Végül összefoglalom az elért eredményeket.

2. fejezet

Emberi preferenciákon alapuló finomhangolási algoritmusok a természetes nyelvfeldolgozásban

2.1. Természetes nyelvfeldolgozással kapcsolatos alapfogalmak

Ebben a fejezetben bevezetem a természetes nyelvfeldolgozás egyes alapfogalmait, illetve a releváns valószínűségi metrikákat. Először a tokenizációt mutatom be, amely a természetes nyelvfeldolgozás egyik alapvető lépése. A tokenizáció során alakítjuk át a szöveget a nyelvi modellek számára kezelhető formátumba, így a modellek működésében fontos szerepet játszik. Emellett kitérek a szövegek vektoros reprezentációjára. Ezután bemutatom azokat a valószínűségi mérőszámokat, amelyek az RLFH és DPO algoritmusoknál kulcsfontosságú szerepet töltenek be (például a veszteségfüggvényben szerepelnek). Ezekre azért van szükség, mert ismeretük elengedhetetlen az RLHF és DPO megértéséhez.

2.1.1. Szövegek reprezentációja a neurális hálókban

2.1.1.1. Tokenizáció

A tokenizáció [8] célja, hogy a nyers szöveget olyan formára alakítsuk, amelyet a gépi modellek képesek feldolgozni. A token a szöveg alapegysége, ami lehet egy szó, szótag vagy akár egyetlen karakter. A természetes nyelvfeldolgozás első lépése a bemeneti szöveg tokenizációja, azaz a szöveg tokenekre bontása.

A hagyományos tokenizáció során a szövegben előforduló tokeneket egy szótárban tároljuk el, ami minden tokenhez egy egyedi numerikus azonosítót rendel. Ez azért fontos, mert a modelleket alkotó neurális hálózatok nem tudnak közvetlenül szöveges adatot kezelni. A szótár segítségével a teljes szöveg átalakítható egy számsorozattá, ami alkalmas a gépi feldolgozásra.

Például, ha egy magyar szöveget tokenizálunk karakterenként, akkor az alábbihoz hasonló szótárat kaphatunk:

!?,.- aábcdeéfghiíjklmnoóöőpqrstuúüűvwxyzAÁBCDEÉFGHIÍJKLMNOÓÖŐPQRSTUÚÜŰVWXYZ

A numerikus azonosító legyen a karakter indexe a listában. Ekkor, ha a szövegünk például az, hogy „csodás nap”, akkor az az alábbi számsorozattá alakítható:

Eredeti: csodás nap

Tokenizált: [9, 30, 23, 10, 7, 30, 5, 22, 6, 27]

Viszont, ha a tokenizációt szavanként végezzük, akkor a szótárunk tartalmazhat teljes szavakat is. Ebben az esetben a szótár az alábbi elemeket tartalmazhatja:

Szó alapú szótár: [„a”, „csodás”, „ez”, „éjszaka”, „Milyen”, „nap”, „reggel”, „szép”, „!”]

Ha ezt a szótárat használjuk a tokenizációra, akkor például a „Milyen csodás ez a nap!” szöveget az alábbi módon alakítjuk át:

Eredeti: Milyen csodás ez a nap!
Tokenizált: [4, 1, 2, 0, 5, 8]

A nyelvi modellek egyik alapvető feladata a tokenek közötti statisztikai mintázatok felismerése, és ezen mintázatok alapján a következő token valószínűségének megbecslése. A modell minden lépésben a bemeneti tokenek alapján próbálja megjósolni a következő tokent, amely a szöveg folytatásában következhet (ez az ún. *next token prediction* feladat).

Matematikailag a modell célja a következő token, t_{n+1} valószínűségének maximalizálása az előző tokenek, $t_1, t_2, \ldots, t_n$ alapján. Ezt a feltételes valószínűséget az alábbi módon fejezhetjük ki:

$$P(t_{n+1} \mid t_1, t_2, \ldots, t_n) \tag{2.1}$$

Ez azt jelenti, hogy a modell megpróbálja kiszámítani, hogy a következő token (t_{n+1}) milyen valószínűséggel következik az előző tokenek sorozata után. A cél az, hogy a modell ezt a valószínűségi eloszlást minél pontosabban becsülje meg, így a szöveg generált folytatása a lehető legtermészetesebb legyen.

A hagyományos természetes nyelvfeldolgozásban a tokenek általában nyelvtani jelentéssel rendelkeznek, például teljes szavak vagy morfémák. Azonban a modernebb, nyelvi modellekhez használt tokenizálók statisztikai alapon állítják elő a tokeneket, ami sokszor értelmetlen szórészleteket (subword) eredményez. Ennek az előnye, hogy megoldást nyújt az ún. Out of Vocabulary (OOV) problémára, ami akkor fordul elő, ha a bemenet egy olyan token, ami nincs benne a szótárban. Mivel a statisztikai alapú tokenizálók általában kisebb részekre bontják a szavakat, a ritkábban előforduló szavakra is tudnak megfelelő tokeneket illeszteni.

A gyakorlatban a Byte-Pair Encoding (BPE) [21] egy gyakran használt tokenizáló algoritmus, amelynek működését egy példa segítségével szeretném bemutatni. Először a szöveget egy elő-tokenizáló (pre-tokenizer) segítségével szavakra bontjuk, valamint az egyes szavak előfordulását is megszámoljuk. A szavakat karakterláncokként reprezentáljuk. Így például az alábbi eredményt kaphatjuk:

(n, a, p, 5), (k, a, p, 4), (a, l, a, p, 1), (n, a, p, o, s, 7), (a, l, a, p, o, s, 3)

A kiinduló szótár a szavak karaktereit tartalmazza, az alábbi módon:

[a, k, l, n, o, p, s]

A BPE algoritmus lényege, hogy megkeressük a leggyakrabban előforduló szimbólum párt, és egyesítjük. Például ebben az esetben az "a", "p" karakterek hússzor követik egymást, így ezekből egy új szimbólumot hozunk létre, amit eltárolunk a szótárban. Az alábbiak szerint módosulnak a szavak és a szótár:

(n, ap, 5), (k, ap, 4), (a, l, ap, 1), (n, ap, o, s, 7), (a, l, ap, o, s, 3)

[a, k, l, n, o, p, s, ap]

A következő lépésben az "o", "s" szimbólumokat egyesítjük, amelyek tízszer követik egymást.

(n, ap, 5), (k, ap, 4), (a, l, ap, 1), (n, ap, os, 7), (a, l, ap, os, 3)

[a, k, l, n, o, p, s, ap, os]

A BPE algoritmus ezeket a lépéseket ismétli, amíg el nem éri a hiperparaméterként megadott szótárméretet, tehát az egyesítések száma előre meghatározható.

Emellett még elterjedt tokenizáló algoritmusok a WordPiece [20] és a SentencePiece [9]. A WordPiece a BPE-hez hasonlóan iteratívan egyesíti a gyakori szimbólum párokat. Azonban, a BPE-vel ellentétben, a szimbólumok egyesítés előtti és utáni valószínűségének hányadosát veszi figyelembe, és azt a párt választja ki, amellyel maximalizálja ezt az értéket. A SentencePiece pedig egy olyan algoritmus, amelynél nem szükséges, hogy a szöveg szóközökkel el legyen választva. Ehelyett egy folyamként kezeli a bemenetet, ami nagyon hasznos azoknál a nyelveknél, ahol nem jellemző a szóközök használata, például a kínai és a japán.

2.1.1.2. Vektorreprezentáció

A Word2Vec algoritmus [14] jelentős változást hozott a nyelvfeldolgozás területén. Bevezette a szavak vektorreprezentációját, vagyis a neurális hálókban a szavakat speciális módon, többdimenziós vektorokként reprezentálja. A legfontosabb újítása az, hogy a szavak vektorreprezentációja szemantikai viszonyokat is képes tárolni. Például vizsgáljuk meg a 2.1 táblázatban található szavak vektorreprezentációját.

Király	**Királynő**	**Férfi**	**Nő**
0.86	0.92	0.74	0.81
0.79	0.89	0.63	0.74
0.68	0.71	0.56	0.60
0.95	0.85	0.77	0.68

2.1. táblázat. Példa a szavak vektorreprezentációjára

Ha a Király vektorból kivonjuk a Férfi vektort, és hozzáadjuk a Nő vektort, akkor megközelítőleg a Királynő vektort kapjuk meg.

$$\vec{v}_{\text{király}} - \vec{v}_{\text{férfi}} + \vec{v}_{\text{nő}} \approx \vec{v}_{\text{királynő}}$$

Ezáltal a nyelvi modellek sokkal jobban képesek kezelni a különböző szavak közötti összefüggéseket. A példában az átláthatóság kedvéért csak kevés dimenziót használtam, de a valóságban az egyes szavakhoz tartozó vektorok akár több száz dimenziósak is lehetnek.

2.1.2. Valószínűségi mérőszámok

A nyelvi modellek célja, hogy a bemeneti tokenek alapján megjósolják a következő tokent. Ehhez elemzik a tokenek közötti összefüggéseket, és ezek alapján egy valószínűségi eloszlást rendelnek a lehetséges következő tokenekhez. Tehát a modell teljesítményét az határozza meg, hogy mennyire pontosan képes előrejelezni a megfelelő tokent.

A nyelvi modellek tanítása során gyakran használják a kereszt-entrópia veszteségfüggvényt, amely a modell által becsült valószínűségi eloszlás (q) és a valós eloszlás (p) közötti

távolságot méri. A tanítás során a célunk, hogy a q eloszlás minél jobban megközelítse a p eloszlást, vagyis hogy minimalizáljuk a kereszt-entrópia értékét.

Definíció 1. Adott X diszkrét valószínűségi változó, amely értékeit az $\mathcal{X}$ halmazon veszi fel $p : X \to [0, 1]$ eloszlás szerint. Ekkor X entrópiája [15]:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{2.2}$$

Az entrópia a rendezetlenség mérőszáma. Ha az eloszlásban sok különböző esemény hasonló valószínűséggel következhet be, akkor az entrópia nagyobb, míg ha az eloszlásban kevés esemény dominál, az entrópia kisebb.

Definíció 2. A kereszt-entrópia értéke p és q diszkrét valószínűségi eloszlások között [15]:

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x) \tag{2.3}$$

ahol:

- $p(x)$ az x esemény valószínűsége a valós eloszlás szerint,
- $q(x)$ az x esemény valószínűsége a modell által becsült eloszlás szerint,
- X az események halmaza, amelyen az eloszlásokat definiáljuk.

A kereszt-entrópia egy aszimmetrikus mérőszám, tehát $H(p, q) \neq H(q, p)$. Az értéke akkor minimális, ha $q(x) = p(x)$ minden x esetén, azaz ha a modell tökéletesen becsli meg a valós eloszlást.

A Kullback-Leibler divergencia (KL divergencia) egy másik gyakran használt mérőszám, amely azt méri, hogy mennyire különbözik a modell által becsült eloszlás a valós eloszlástól.

Definíció 3. A Kullback-Leibler divergencia (KL divergence) értéke p és q diszkrét valószínűségi eloszlások között [15]:

$$D_{\mathrm{KL}}(p \| q) = \sum_{x \in \mathcal{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right). \tag{2.4}$$

A kereszt-entrópia és a Kullback-Leibler divergencia közvetlenül összefügg egymással, ugyanis

$$D_{\mathrm{KL}}(p \| q) = H(p, q) - H(p) \tag{2.5}$$

ahol:

- $H(p, q)$ a kereszt-entrópia p és q között,
- $H(p)$ pedig a p eloszlás entrópiája.

2.2. Megerősítéses tanulás emberi visszajelzés alapján

A megerősítéses tanulás emberi visszajelzés alapján, vagyis Reinforcement Learning from Human Feedback (RLHF) [22] megvalósítása az alábbi lépésekből áll:

- nyelvi modell előtanítása (pretrain)
- jutalommodell (reward model) tanítása
- nyelvi modell finomhangolása megerősítéses tanulással (reinforcement learning)

Ezekből szigorúan véve csak a második és harmadik lépés tartozik az RLHF-hez. Azonban az InstructGPT [16] esetén a nyelvi modell előtanítását is az RLHF lépései közé sorolták, így a következőekben mindegyiket részletesen bemutatom.

2.2.1. Nyelvi modell előtanítása

Az RLHF-hez szükségünk van egy transzformer architektúrájú, next token prediction-re tanított nyelvi modellre, amely kiinduló modellként szolgál. Az előtanítás során a modellt hatalmas mennyiségű címkézetlen szöveges adaton tanítják meg a nyelv szerkezetére. Ilyenkor a modell még nem egy specifikus feladatra van optimalizálva, hanem egy általános, széleskörű tudás megszerzése a cél, ami megalapozza a későbbi lépéseket. A kiinduló modellt mi is előtaníthatjuk, de gyakori egy már előtanított modell használata. Például az OpenAI a GPT-3 egy kisebb verzióját használta az InstructGPT-hez, az Anthropic pedig 10 millió és 52 milliárd közötti paraméterrel rendelkező transzformer modelleket [10].

Az előtanított modellt finomhangolhatjuk egy specifikus adathalmazon. A felügyelt finomhangolás (Supervised fine-tuning, SFT) egy célzottab tanítási folyamat, ugyanis egy konkrét feladatra készítjük fel a modellt, egy olyan címkézetlen adathalmaz segítségével, ami a specifikus feladathoz kapcsolódik. Ez a lépés opcionális, de jelentősen javíthatja a modell teljesítményét. Az előző példánál maradva, az InstructGPT preferált emberi szövegek alapján, az Anthropic RLHF modellje pedig a 3H, vagyis „helpful, honest and harmless" kritérium alapján volt finomhangolva [10].

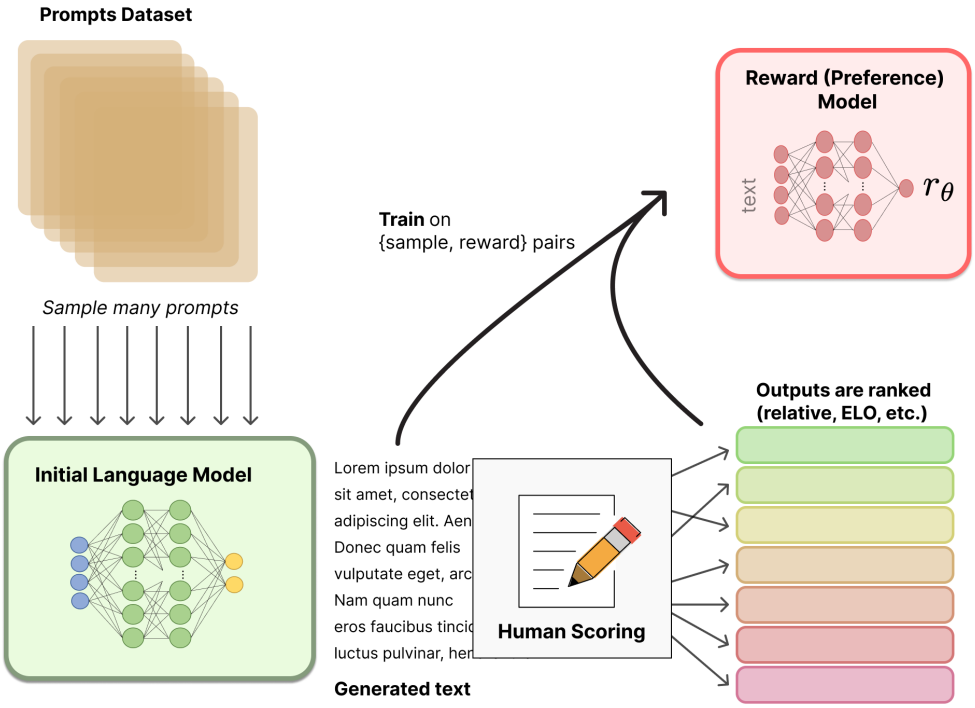
2.2.2. Jutalommodell tanítása

A következő lépés a modellünk tanítása az emberi visszajelzések alapján. Ehhez azonban arra van szükségünk, hogy emberek értékeljék a modell különböző válaszait, amihez mintát kell vennünk a modellből. Tegyük fel, hogy a modell minden s mintájára kapunk egy skaláris emberi értékelést / jutalmat annak minőségéről. Jelölje ezt $R(s)$. Ekkor, ha $R(s_1) > R(s_2)$, az azt jelenti, hogy az s_1 minta jobb az s_2-nél. Tehát a célunk a minták várható jutalmának maximalizálása.

Azonban az emberi jutalmak beszerzésénél több probléma is felmerülhet. Az egyik, hogy a gyakorlatban a Human-in-the-loop megközelítés nagyon költséges és időigényes. A tanítás iterációin keresztül a különböző minták emberek általi kiértékelése valószínűleg olyan körülményes és hosszadalmas lenne, hogy a befektetés nem térülne meg. Erre a megoldás az lehet, hogy ahelyett, hogy folyamatosan emberekkel értékeltetnénk ki az összes mintát, korlátozott számú emberi értékelés alapján építünk egy modellt, ami a későbbiekben az emberek helyett használható a minták kiértékelésére. Ezt a modellt nevezik jutalommodellnek.

Viszont ehhez is szükség van emberek által kiértékelt mintákra, ami elvezet a következő problémához: egy szöveg minőségének megítélése szubjektív, az emberek véleménye sokszor nem egyértelmű, ha konkrét számértékekkel kell pontozniuk a mintákat. Ezért egy darab minta önálló pontozása helyett az emberek feladata az, hogy mintapárokat hasonlítsanak össze, és kiválasszák a jobbat. Az így előállított preferenciákból ki lehet építeni egy rendszert valamilyen stratégia (pl. ELO algoritmus [6]) alapján, amely lehetővé teszi, hogy egyértelmű skaláris értékeket rendeljünk az egyes mintákhoz. A jutalommodellt ezen értékek alapján tudjuk tanítani, biztosítva ezzel, hogy a modell hatékonyan becsülje meg

az egyes minták minőségét, és megfelelően reprezentálja az emberi preferenciákat. Ezeket a lépéseket jól szemlélteti a 2.1 ábra.



2.1. ábra. A jutalommodell tanítás lépéseinek áttekintése [10]

Az egyik vizsgált kutatás során a feladat egy x szöveg összefoglalása (summarization) volt [22]. Ekkor a modellt arra tanítják, hogy megbecsülje, melyik $y \in \{y_0, y_1\}$ összegzés jobb az emberek szerint, megadott x alapján. Tegyük fel, hogy y_i az emberek által preferált összegzés, ekkor a veszteségfüggvény az alábbi módon írható fel [22]:

$$\text{loss}(r_\theta) = -E_{(x,y_0,y_1,i)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_i) - r_\theta(x,y_{1-i})\right)\right)\right] \tag{2.6}$$

ahol $r_\theta(x,y)$ az x szöveg y összegzésének a jutalommodell által becsült értéke, θ paraméterekkel, és D az emberi értékeléseket tartalmazó adatbázis.

A cél az, hogy az $r_\theta(x,y_i) - r_\theta(x,y_{1-i})$ különbség minél nagyobb legyen, mivel ez azt jelenti, hogy a jutalommodell a preferált választ tartja jobbnak. A sigmoid függvény ezt az értéket a következő módon alakítja át a $[0,1]$ tartományba:

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{2.7}$$

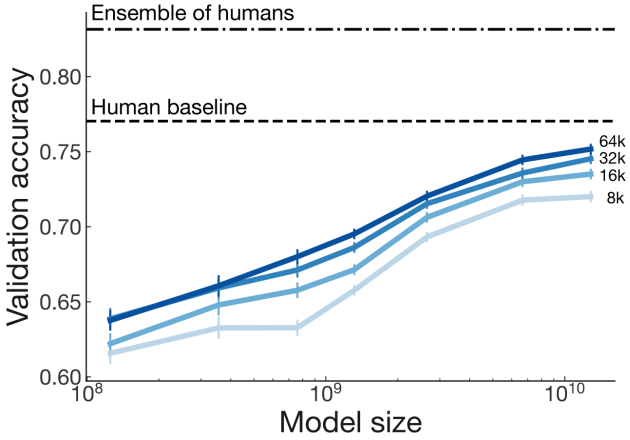
így a különbség valószínűségként értelmezhető. Minél nagyobb a különbség, a sigmoid függvény kimenete annál közelebb lesz 1-hez. Ha a különbség 0, akkor a kimenet 0.5, tehát a modell bizonytalan. Ha a nem preferált választ értékeli jobbnak a modell, akkor a különbség negatív értéket vesz fel. Minél negatívabb a különbség, a sigmoid kimenete annál közelebb lesz 0-hoz.

Ennek a valószínűségi értéknek vesszük a logaritmusát, és megszorozzuk -1-gyel. Ekkor:

- ha a valószínűség 1, akkor a függvény értéke $-1 \cdot log(1) = 0$, azaz a veszteség minimális.
- ha a valószínűség 0, akkor a függvény értéke $-1 \cdot log(0) = -1 \cdot (-\infty) = \infty$-hez tart, azaz a veszteség maximális.

Tehát minél jobbra értékeli a preferált válaszokat a modell, a veszteségfüggvény értéke annál kisebb lesz.

Felmerülhet, hogy az így alkotott jutalommodell valóban megfelelő becslést tud-e adni az egyes minták minőségének értékére. A 2.2 ábrán látható, hogy a kellően nagy, 6,7 milliárd paraméterrel rendelkező modell, bár a több emberből álló csoport pontosságát nem éri el, de egy ember teljesítményét megközelíti (human baseline).



2.2. ábra. Jutalommodell pontossága az adatok és a modell méretének függvényében [22]

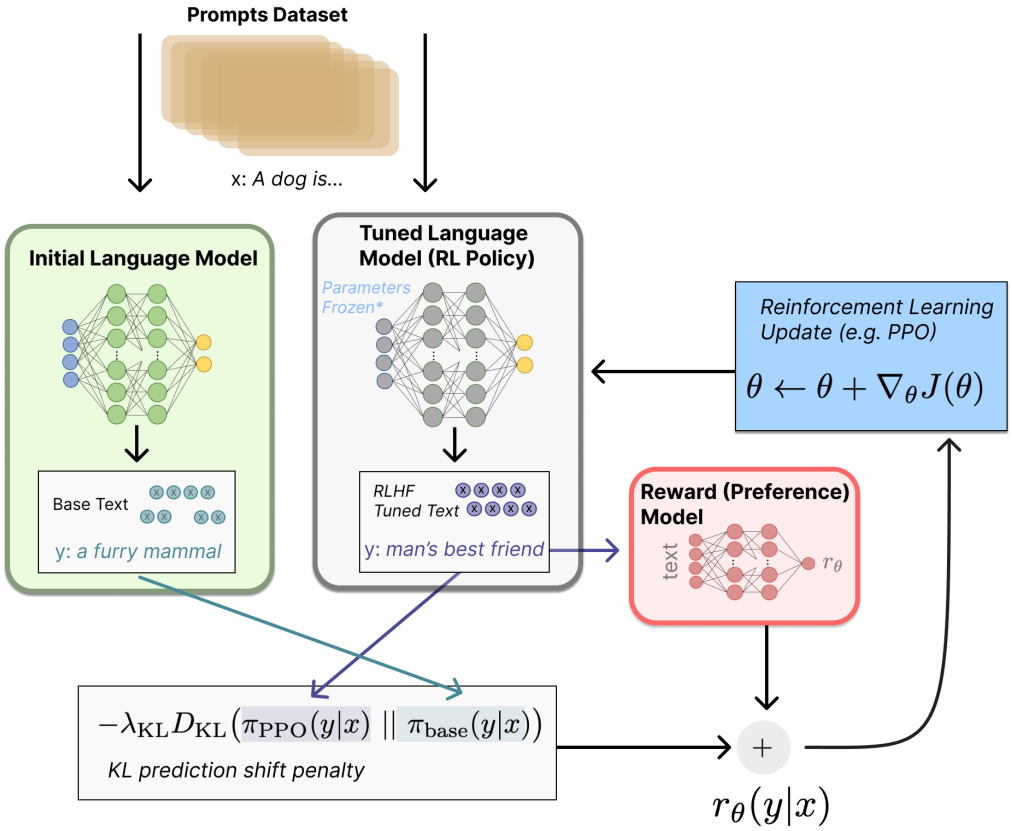
2.2.3. Nyelvi modell finomhangolása megerősítéses tanulással

Az utolsó lépésben a jutalommodellt felhasználva finomhangoljuk a modellt, hogy az emberi preferenciákat figyelembe véve jobb minőségű kimeneteket adjon. Ehhez megerősítéses tanulást (Reinforcement Learning, RL) alkalmazunk, amely során a modellnek egy új stratégiát (policy) tanítunk. A folyamatot a 2.3 ábra mutatja be, amelynek a lépéseit a következőekben részletesen elemzem.

A kiinduló nyelvi modellből (Initial Language Model, π_{base}) készítünk egy másolatot, aminek a paramétereit finomhangolni szeretnénk (Tuned Language Model, π_{PPO}). Azonban a nyelvi modelleknek akár több milliárd paramétere is lehet, így a tanításuk nagyon költséges. Ezért a paraméterek egy részét lefagyasztjuk (freeze), vagyis nem frissítjük a tanítás közben.

A tanítás során mintát veszünk a tanítandó modellből: bemenetként promptokat (x) adunk a modellnek, amelyek alapján kimeneteket (y) generál. Ezeket értékeli a jutalommodell ($r_\phi(y|x)$). Az cél az, hogy a kimenetek értékelése minél magasabb legyen, mivel ez azt jelenti, hogy jól illeszkednek az emberi preferenciákhoz. Azonban ha a stratégiát kizárólag a jutalommodell alapján frissítjük, akkor előfordulhat, hogy a modell kimenetei túlságosan elkezdenek a jutalommodellhez illeszkedni, és ez akár a teljesítmény romlásához is vezethet: a modell értelmetlen, halandzsa szöveget kezd generálni.

Ahhoz, hogy ezt elkerüljük, arra is figyelünk, hogy a tanítandó modell ne távolodjon el túlságosan a kiinduló modelltől. Ehhez a Kullback-Leibler (KL) divergenciát használjuk, amely a két modell kimeneteinek valószínűségi eloszlásai közötti különbséget méri.



2.3. ábra. Finomhangolás megerősítéses tanulással lépéseinek áttekintése [10]

Tehát a megerősítéses tanulás jutalomfüggvényét (R, reward function) az alábbi módon állítjuk össze: vesszük a jutalommodell értékelését a tanítandó LM y kimenetére x prompt függvényében, ezt jelöli $r_\theta(x, y)$. Emellett negatív előjellel hozzávesszük a modellek kimeneteinek Kullback-Leibler (KL) divergenciáját (D_{KL}), így büntetjük, ha az új stratégia túlságosan eltávolodik az eredetitől. A büntetés mértékét a β paraméterrel tudjuk szabályozni. Tehát a jutalomfüggvény az alábbi módon írható fel [22]:

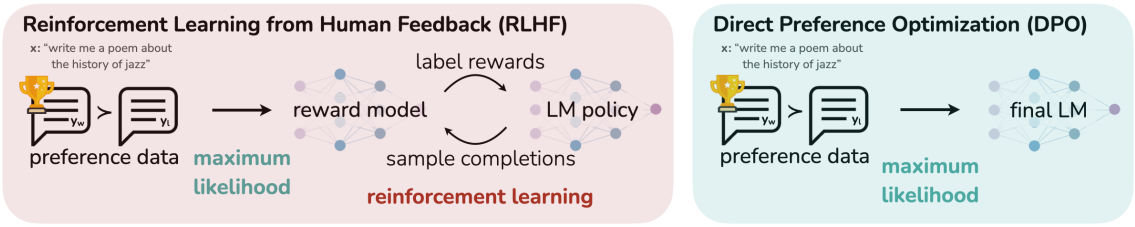
$$R(x, y) = r_\theta(x, y) - \beta \cdot D_{\text{KL}}(\pi_{PPO}(y|x) \| \pi_{base}(y|x)) \tag{2.8}$$

Végül az így kapott jutalomfüggvény alapján frissítjük a stratégiát Proximal Policy Optimization (PPO) [19] segítségével. A PPO egy megerősítéses tanulásos algoritmus, amelynek célja a stratégia (policy) tanításának stabilizálása a frissítés mértékének korlátozásával. Több okból is fontos, hogy elkerüljük a drasztikusan nagy stratégia frissítéseket: egyrészt a kisebb frissítések nagyobb eséllyel konvergálnak az optimális megoldáshoz, másrészt egy túl nagy lépés esetén a stratégia jelentősen eltérhet az optimumtól, ami „off the cliff” eredményhez vezethet. Ezután hosszú időbe telhet a regenerálódás, vagy akár nem is lehetséges.

Az RLHF nagy előnye, hogy az emberi preferenciákat közvetlenül modellezi. Azonban több hátránya is van: nagy számításigényű (három LM), a jutalommodell megfelelő betanítása nehéz lehet, és az emberi preferenciák nem megbízhatóak, így fennáll a reward hacking veszélye.

2.3. Közvetlen preferenciaoptimalizálás

Bár a közvetlen preferenciaoptimalizálás (Direct Preference Optimization, DPO) [18] célja az RLHF-hez hasonlóan az emberi preferenciák beépítése a nyelvi modell tanításába, a feladat megközelítésében sok különbség van. A fő eltérést a 2.4 ábra szemlélteti: a DPO során nincs szükség sem explicit jutalommodell elkészítésére, sem megerősítéses tanulásra. Ehelyett a DPO közvetlenül optimalizálja a nyelvi modellt az emberi preferenciák alapján.



2.4. ábra. Különbség az RLHF és a DPO között [18]

2.3.1. Adatok előkészítése

A DPO megvalósításához először is szükségünk van az emberi preferenciákat tükröző adatokra, amelyeket speciális módon kell előkészíteni. Adott x promptra két osztályba soroljuk az y válaszokat:

- y_w: az emberek által preferált / jó válasz, és
- y_l: a nem preferált / rossz válasz.

Tehát itt nincs szükség arra, hogy skaláris értékeket rendeljünk minden egyes mintához, mint az RLHF esetén.

Az adatokat azért kell ilyen módon előkészíteni, mert a DPO az eredeti feladatot egy bináris osztályozási feladatra vezeti vissza, vagyis a szöveget a jó / rossz kategóriákba sorolja. A cél az, hogy úgy tanítsuk a modellt, hogy a preferált válaszok valószínűsége nagyobb legyen, mint a nem preferált válaszoké.

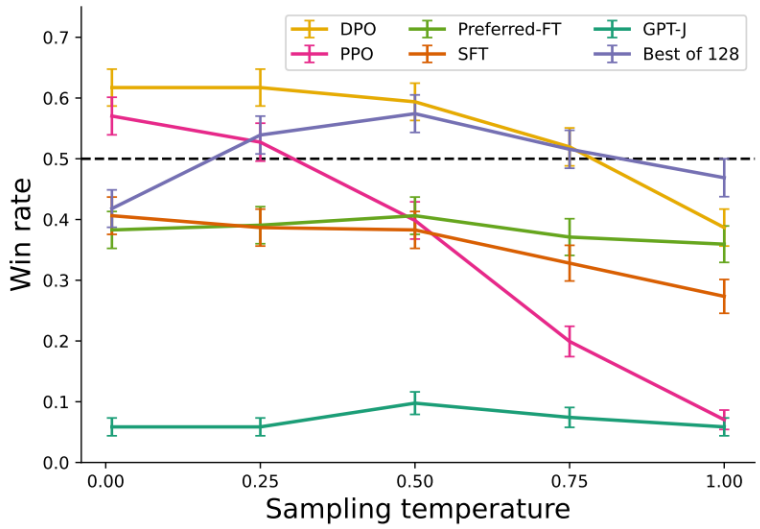
2.3.2. Optimalizálás a preferenciák alapján

A tanítás során a jutalom maximalizálására törekszünk KL-divergencia korlátozás mellett, ami biztosítja, hogy a finomhangolt modell (π_θ) ne távolodjon el túlságosan a referencia modelltől (π_{ref}). Az optimalizáláshoz bináris kereszt-entrópiát (binary cross-entropy) használunk veszteségfüggvénynek. A modell súlyainak frissítése során azt szeretnénk elérni, hogy növeljük a preferált válaszok relatív logaritmikus valószínűségét a nem preferált válaszokhoz képest. Ekkor a veszteségfüggvény gradiense az optimalizálandó paraméterre az alábbi módon írható fel [18]:

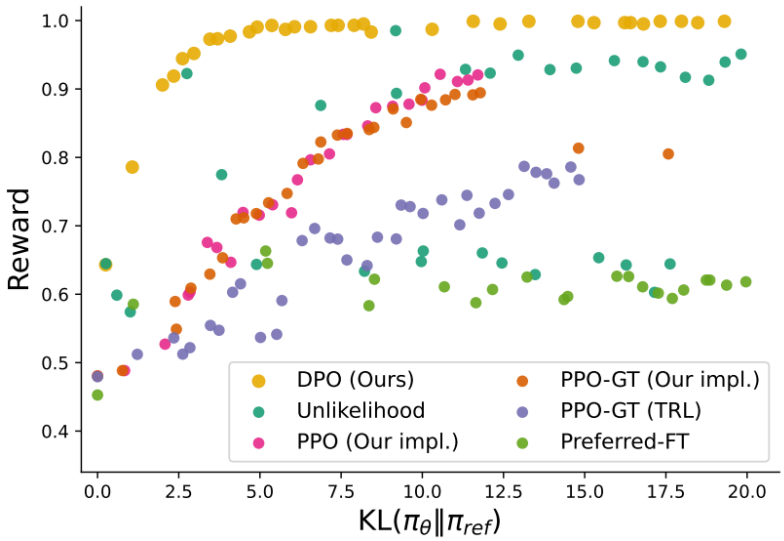
$$\begin{aligned} &\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = \\ &-\beta E_{(x,y_w,y_l)\sim D}\Big[\sigma\left(r_\theta(x,y_l) - r_\theta(x,y_w)\right)\left(\nabla_\theta \log \pi(y_w \mid x) - \nabla_\theta \log \pi(y_l \mid x)\right)\Big] \end{aligned} \tag{2.9}$$

A DPO tanítás során nincs szükség sem explicit jutalommodell, sem megerősítéses tanulás használatára. Ez egyszerűbbé teszi az implementációt, valamint a számításigényt is jelentősen csökkenti.

Felmerülhet, hogy a DPO egyszerűbb, kisebb számításigényű megoldása fel tudja-e venni a versenyt az RLHF-fel. Ehhez összehasonlították a két algoritmust, méghozzá úgy, hogy figyelembe vették az elért jutalom mellett a KL értéket is, ugyanis a nagyobb jutalom magas KL érték mellett nem feltétlenül kívánatos (ez azt jelenti, hogy a modell túlságosan eltávolodott az eredeti modelltől). Az egyik kísérlet során Reddit posztok összefoglalása volt a feladat, és a különböző algoritmusok válaszait GPT-4 vetette össze az emberek által írt összefoglalókkal. A 2.5 ábrán látható az algoritmusok nyerési aránya az amberek által írt összefoglalókkal szemben. Látható, hogy a DPO a PPO legjobb teljesítményénél is jobban teljesít, és robusztusabb a sampling temperature változására. Egy másik kísérlet során IMDb értékelésekből kellett pozitív hangvételű értékeléseket generálni. A 2.6 ábrán pedig megfigyelhető, hogy a DPO éri el a legmagasabb várható jutalmat az összes KL értékre. [18] Ezek alapján kijelenthetjük, hogy a DPO legalább olyan jól teljesít, mint az RLHF.



2.5. ábra. Nyerési arányok az emberek által írott összefoglalókkal szemben, GPT-4 által értékelve [18]



2.6. ábra. A generált IMDb értékelés várható jutalma a KL függvényében [18]

3. fejezet

Célkitűzések

A Reinforcement Learning from Human Feedback és a Direct Preference Optimization összehasonlítása során egyértelművé váltak az utóbbi módszer előnyei, így a DPO implementációja tűnt célszerűbbnek. A dolgozatommal az a célom, hogy megvizsgáljam, milyen mértékben lehetséges DPO alkalmazásával javítani a nyelvi modellek teljesítményén. Ehhez először angol nyelven végzem el a DPO alapú finomhangolást egy kisebb modellen, hogy a gyakorlatban is teszteljem a módszer működését. A fő célom pedig a magyar nyelvre való áttérés, és a megvalósítás skálázása, vagyis egy nagyobb modell használata a finomhangoláshoz.

Ehhez az alábbi feladatokat szükséges teljesítenem:

- **Direct Preference Optimization megvalósítása angol nyelven**
  - Angol nyelvű adatok gyűjtése
    - Angol nyelvű adathalmaz gyűjtése a felügyelt finomhangoláshoz (SFT), amennyiben szükséges (opcionális lépés)
    - Angol nyelvű adathalmaz gyűjtése a Direct Preference Optimization finomhangoláshoz
    - Kis méretű kiinduló modell meghatározása (milliós nagyságrendű paraméterszám)
  - Direct Preference Optimization megvalósítása
  - A finomhangolt modell tesztelése, eredmények értékelése
- **Direct Preference Optimization megvalósítása magyar nyelven**
  - Magyar nyelvű adatok gyűjtése
    - Magyar nyelvű adathalmaz gyűjtése a felügyelt finomhangoláshoz (SFT), amennyiben szükséges (opcionális lépés)
    - Magyar nyelvű adathalmaz gyűjtése a Direct Preference Optimization finomhangoláshoz
    - Nagy méretű kiinduló modell meghatározása (milliárdos nagyságrendű paraméterszám)
  - Direct Preference Optimization megvalósítása
  - Teszt kérdéssor összeállítása a kiinduló és a finomhangolt modell válaszainak összehasonlításához
  - A finomhangolt modell tesztelése, eredmények értékelése

4. fejezet

Rendszerterv

4.1. A Transformer Reinforcement Learning könyvtár finomhangolási algoritmusai

A DPO-hoz lehetséges akár saját implementációt is készíteni, viszont léteznek olyan könyvtárak, amelyekben a szükséges algoritmusok már elő vannak készítve, így praktikusabb ezeket használni. Az egyik legelterjedtebb a Huggingface Transformer Reinforcement Learning (TRL) könyvtára [25], ami a transzformer architektúrájú nyelvi modellek finomhangolására lett kifejlesztve, így először ezt vizsgáltam meg. A TRL-ben több kulcsfontosságú finomhangolási algoritmus is implementálva van, amelyek az RLHF-hez és a DPO-hoz is hasznosak. Az algoritmusok különböző Trainer osztályokon keresztül érhetők el, amelyek közül az alábbiak a legrelevánsabbak:

- **SFTTrainer**: Felügyelt finomhangolás / Supervised Fine-tuning (SFT)
- **RewardTrainer**: Jutalommodell / Reward Modeling (RM)
- **PPOTrainer**: Proximal Policy Optimization (PPO)
- **DPOTrainer**: Direct Preference Optimization (DPO)

A DPO finomhangolás implementációjához két komponens használható fel: a felügyelt finomhangoláshoz (ami opcionális lépés) az SFTTrainer, és a DPO finomhangoláshoz a DPOTrainer.

4.2. Memóriahatékony finomhangolás Parameter-Efficient Fine-Tuning segítségével

Az implementáció során kihívást jelentett, hogy a felügyelt finomhangolás és a DPO finomhangolás futtatása is nagy memóriaigényű. Ezért olyan módszereket kerestem, amelyekkel a memóriahasználat csökkenthető, hogy a rendelkezésemre álló erőforrásokkal is sikeresen lefusson a finomhangolás. Az áttörést a Parameter-Efficient Fine-Tuning (PEFT) [26] bevezetése hozta el, amely a teljes finomhangolás helyett csak a paraméterek egy részét módosítja.

A PEFT algoritmusok a nagy nyelvi modellek finomhangolásának optimalizálására lettek kifejlesztve. A nagy nyelvi modellek finomhangolása során az összes paraméter módosítása hatalmas memóriaigényű lehet, ami a hatékonyság rovására mehet. A PEFT technikák célja, hogy csökkentsék a finomhangolt paraméterek számát, és ezáltal a finomhangoláshoz szükséges memória- és számításigényt, miközben a modell a teljes finomhangolást

megközelítő teljesítményt nyújt. Ehhez az összes paraméter helyett csak a paraméterek egy részhalmazát frissítik, a többin pedig nem változtatnak. Ezzel jelentős mennyiségű memória és idő takarítható meg, így a PEFT technikák lehetővé teszik, hogy korlátos erőforrásokkal is finomhangolhatóak legyenek a nagy nyelvi modellek.

Több PEFT módszer is létezik, amelyek mind más megközelítést használnak a finomhangolandó paraméterek számának csökkentéséhez. A módszerek típusait és fejlődését a 4.1 ábra foglalja össze, amelynél az alábbi 5 kategóriát különböztetjük meg:

- **Additive Fine-tuning:** új paramétereket ad hozzá a modellhez, amelyeket finomhangolunk, míg a modell eredeti paraméterei változatlanok maradnak. Ezzel lehetővé teszi a modell számára, hogy rugalmasan alkalmazkodjon az új feladatokhoz, anélkül, hogy az eredeti beállításait módosítaná.
- **Unified Fine-tuning:** egységes keretrendszert biztosít a finomhangoláshoz, amely megkönnyíti a különböző finomhangolási módszerek integrálását egy egységes architektúrába.
- **Reparameterized Fine-tuning:** alacsony rangú transzformációkat alkalmaz a modell tanítható paramétereinek csökkentésére. Például a paraméterek változásait alacsony rangú mátrixokban tárolja el.
- **Hybrid Fine-tuning:** ötvözi a különböző PEFT módszereket, hogy kihasználja azok előnyeit. Általában a hibrid módszerrel a modell összességében jobb teljesítményt nyújt, mint az egyes PEFT módszerekkel külön-külön.
- **Partial Fine-tuning:** kiválasztja a paraméterek egy részhalmazát, amelyek kritikusak a feladatok szempontjából, és ezeket finomhangolja, a kevésbé fontos paramétereket pedig figyelmen kívül hagyja.

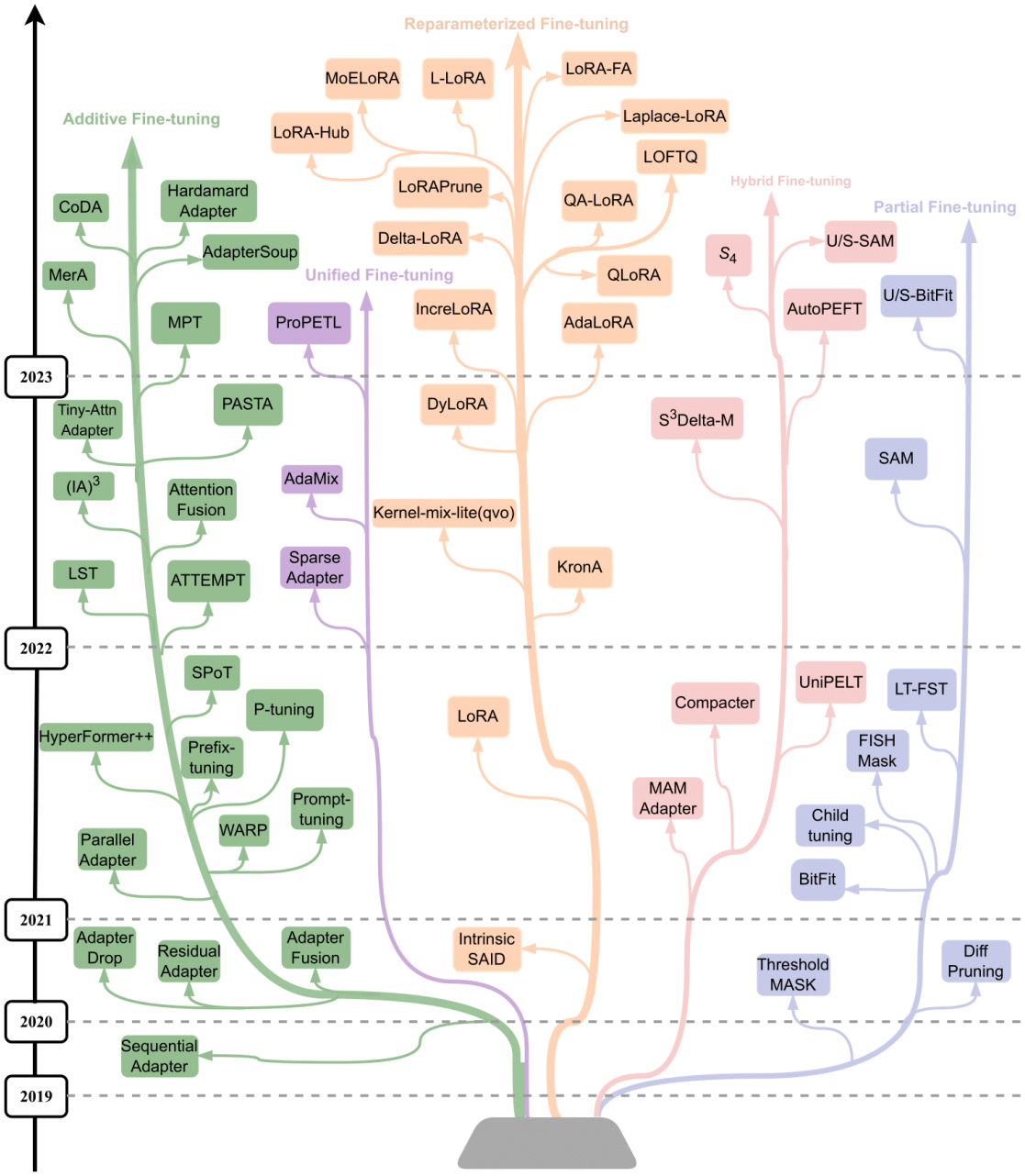
Ezek közül a Low-Rank Adaptation (LoRA) [7] mellett döntöttem, amely a Reparameterized Fine-tuning módszerek közé tartozik. A LoRA alapötlete azon alapszik, hogy a nagy nyelvi modellek paraméterei gyakran redundáns információkat tartalmaznak, így az összes paraméter tárolása helyett elegendő lehet a modell súlyaira egy alacsony rangú becslést adni. Ennek érdekében a finomhangolás során az eredeti paramétereket „lefagyasztjuk", tehát változatlanul hagyjuk, és a változásokat két alacsony rangú mátrix segítségével tároljuk el. A finomhangolt súlyokat az alábbi módon kaphatjuk meg [7]:

$$W = W_0 + \Delta W = W_0 + BA, \quad \text{ahol:} \tag{4.1}$$

- W: a finomhangolás eredményeképpen kapott súlymátrix,
- $W_0 \in \mathbb{R}^{d \times k}$: az előtanított modell eredeti súlyai (pre-trained weight matrix),
- ΔW: a finomhangolás során kapott súlyváltozások,
- $B \in \mathbb{R}^{d \times r},\ A \in \mathbb{R}^{r \times k}$: alacsony rangú mátrixok,
- $r \ll \min(d, k)$

Tehát a súlyok változását két alacsony rangú mátrix szorzataként kapjuk meg. Ezáltal egy közelítést adunk a paraméterek értékére, amivel jelentősen, akár tízezred részére csökkenthetjük a tárolandó paraméterek számát, így a modell finomhangolása kevesebb memóriát igényel (akár harmadára csökken a GPU memóriaigény) és gyorsabban végrehajtható. [7]

A Huggingface PEFT könyvtárában [12] implementálva van a LoRA, így egyszerűen integrálható a finomhangolási folyamatba.



4.1. ábra. A különböző PEFT algoritmusok fejlődése az elmúlt évek során. [26]

5. fejezet

Közvetlen preferenciaoptimalizálás angol nyelven

Az első DPO finomhangolást egy kisebb modellen végeztem, angol nyelven, mivel csak korlátos erőforrások álltak rendelkezésemre. Ebben a fejezetben részletezem az implementáció megtervezését és elkészítését.

5.1. Adatbázis

A felügyelt finomhangolás és a DPO implementációjához az alábbiakra volt szükségem:

- egy kiinduló modell,
- egy adathalmaz a felügyelt finomhangoláshoz (SFT),
- egy adathalmaz a DPO finomhangoláshoz.

A Huggingface Models és Datasets oldalán rengeteg nyílt adathalmaz elérhető, így ezek közül válogattam. Az erőforrások korlátossága miatt igyekeztem olyan modellt és olyan adatokat találni, amelyek mérete nem gátolja a tanítás hatékony futtatását. Ezért egy kompromisszumos megoldás mellett döntöttem, és kiinduló modellnek a Facebook OPT 350M modellt választottam [27]. A 350 millió paraméterével ez egy viszonylag kis modell, amely a Meta által fejlesztett Open Pre-trained Transformer (OPT) modellcsalád része. Az OPT nagyrészt angol nyelven lett tanítva, és a GPT-3-hoz hasonló architektúrával rendelkezik.

A felügyelt finomhangoláshoz egy pénzügyi adathalmazt [4] használtam, amely tartalmazza a Stanford Alpaca [23] adathalmazt, a FiQA [11] adathalmazt és további 1300, GPT-3.5 által generált adatot. A DPO finomhangoláshoz egy matematikai feladatokat tartalmazó adathalmazt [2] használtam. Ez a DPO-nak megfelelő formátumban tartalmazza az adatokat, összesen 2420 sorból áll, ebből egy példa az alábbi:

> **Feladat:** How can I simplify the algebraic expression ‘(3xˆ2 - 4yˆ3) / (2x)‘?. Take a deep breath, think step by step, and give an accurate response.
>
> **Választott:** To simplify the algebraic expression ‘(3xˆ2 - 4yˆ3) / (2x)‘, we can follow a few steps: Step 1: Distribute the division symbol by multiplying the expression by the reciprocal of the denominator. The reciprocal of ‘2x‘is ‘1/(2x)‘, so the expression becomes ‘(3xˆ2 - 4yˆ3) * (1/(2x))‘. Step 2: Simplify within the parentheses by dividing each term separately. - For the first term, ‘3xˆ2‘, divide ‘3xˆ2‘by ‘2x‘. This gives us ‘(3xˆ2) / (2x) = (3/2) * (xˆ2 / x) = (3/2) * x‘. - For the second term, -4yˆ3‘, divide -4yˆ3‘by ‘2x‘. This gives

us ‘(-4y^3) / (2x) = (-2) * (y^3 / x)‘. Step 3: Combine the simplified terms from Step 2. The expression now becomes ‘(3/2) * x - 2 * (y^3 / x)‘. So, the simplified form of the algebraic expression ‘(3x^2 - 4y^3) / (2x)‘is ‘(3/2) * x - 2 * (y^3 / x)‘.

Elutasított: To simplify the algebraic expression ‘(3x^2 - 4y^3) / (2x)‘, you can divide each term in the numerator by the denominator. First, let’s divide ‘3x^2‘by ‘2x‘. Since both terms have a common factor of ‘x‘, we can simplify this expression to ‘3x‘. Next, we divide -4y^3‘by ‘2x‘. We can simplify this expression by dividing each term separately. Dividing -4‘by ‘2gives -2‘. Then, dividing ‘y^3‘by ‘xgives ‘y^3/x‘. So, the simplified form of ‘(3x^2 - 4y^3) / (2x)‘is ‘3x - 2y^3/x‘.

5.2. Finomhangolás implementációja

Miután előkészítettem az implementációhoz szükséges eszközöket és adatokat, továbbléphettem a gyakorlati megvalósításra. A megvalósítást Python nyelven, Google Colab környezetben végeztem. Az első lépés a kiinduló modell, vagyis a Facebook OPT 350M modell betöltése volt, amihez külön készítettem egy konfigurációt BitsAndBytesConfig segítségével. Ebben a konfigurációban 4 bites kvantálást állítottam be, ami azt jelenti, hogy a modell paramétereit 4 bites formátumban tárolom el, így azok kevesebb tárhelyet igényelnek. A kvantálás elősegíti az adatok tömörítését és a memóriahasználat csökkentését.

A kiinduló modell betöltését a transformers könyvtár AutoModelForCausalLM osztályának from_pretrained függvényének segítségével végeztem el. A paraméterekben összekapcsoltam a modellt a létrehozott kvantálási konfigurációval, emellett beállítottam, hogy a modellhez automatikusan legyenek hozzárendelve a rendelkezésre álló hardvereszközök. Emellett betöltöttem a modellhez tartozó tokenizálót is a transformers AutoTokenizer osztályának segítségével.

A LoRA megvalósításához a Huggigface PEFT könyvtárának LoraConfig osztályát használtam. Az r értékét, vagyis a mátrixok rangját 16-ra állítottam.

A következő lépés a felügyelt finomhangolás, vagyis az SFT végrehajtása volt. Az adatok betöltéséhez a Huggingface Datasets könyvtárának load_dataset függvényét használtam. Az adathalmazból az első tízezer sort használtam fel a finomhangoláshoz, és a train_test_split függvény segítségével szétbontottam tanító és teszt halmazokra 7:3 arányban. Az SFT megvalósításához pedig az SFTTrainer osztályt használtam, és itt beállítottam többek között a kiinduló modellt, az adathalmazokat, a tokenizert és a LoRA konfigurációt.

Végül a DPO megvalósítása következett, amihez a DPOTrainer osztályt használtam. A szükséges adatokat az SFT adatokhoz hasonlóan töltöttem be, és a LoRA konfiguráció is megegyezett az SFT esetében használt változattal. Az adathalmaz 2418 sort tartalmazott, és 80%-át tanító, 20%-át pedig teszt halmazba osztottam. A teljes forráskód megtekinthető az F.1 függelékben található GitHub repository-ban, az angol_dpo.py fájlban.

5.3. Kiértékelés

Az SFT tanítás 4 epoch-on keresztül ment végig, összesen 3500 step (lépés) során. A training loss (tanítási adathalmazon vett veszteség) és a validation loss (validációs adathalmazon vett veszteség, ami ebben az esetben a teszt adathalmaz) az 5.1 táblázatban látható módon csökkent a megtett lépések előrehaladtával. Az első 1500 lépésnél jelentős javulás ment végbe, utána a változás egyre csökkent, a 3000. és 3500. step között már kevesebb, mint egy százados a javulás.

Step	Training Loss	Validation Loss
500	3,9439	3,6326
1000	3,6421	3,5552
1500	3,5487	3,5071
2000	3,5289	3,4693
2500	3,4705	3,4450
3000	3,4312	3,4289
3500	3,4279	3,4225

5.1. táblázat. Az SFT finomhangolás során a training loss és a validation loss változása a megtett step-ek függvényében.

A DPO tanítás 2 epoch-ban valósult meg összesen 604 step során. Az 500. step-nél a training loss 0,8499 és a validation loss 0,7197 volt, a tanítás végére a training loss elérte a 0,8343-at. A loss értékek elemzése után a modell éles tesztelése következett. A következőekben bemutatom a kezdeti modell (Alap modell), valamint a felügyelt finomhangolás és DPO finomhangolás utáni modell (DPO modell) válaszait néhány egyszerű promptra.

Az 5.2 táblázat tartalmazza a modellek válaszát egy csevegő stílusú kérdésre. Az alap modell válasza inkoherens és irreleváns a prompt szempontjából. Ezzel szemben a DPO modell első mondatában az alapvető társalgási normáknak megfelelően válaszol, majd emberi módon kezd megosztani magáról információkat, de a mondat második részében nem teljesen egyértelmű, hogy mi „not enough" és az „it" mire utal, valószínűleg a fizetés nem elég a magas tandíjhoz.

Az 5.3 táblázatban látható matematikai feladatot egyik modell sem tudta megoldani, de mindkettő a kontextusnak megfelelően válaszolt, az alap modell itt egy fokkal eredetibb volt, mivel azt mondta, van számológépe, de összességében úgy gondolom, a két modell nagyjából egyformán teljesített ebben az esetben.

Az 5.4 táblázatban található esetben egy humoros választ várunk el, viszont az alap modell válasza teljesen irreleváns. Reddit-ről említés sem volt a kérdésben, összességében a válaszának semmi köze a prompthoz. A DPO modell sem teljesített sokkal jobban, megfigyelhető a hallucináció a válaszában, ugyanis ezek az idézetek nem léteznek, de legalább a megemlített személyek humoros műfajokhoz kapcsolódnak.

Az 5.5 táblázatban befektetési tanácsot kérünk a modellektől. Az alap modell itt teljesen értékelhetetlen választ adott, mivel visszakérdezett, emellett a regisztert sem sikerült eltalálnia, mivel egy befektetési kérdés esetén a 'total loser' egyáltalán nem illik a szövegbe. A DPO modell itt egyértelműen jobban teljesített, bár nem adott konkrét választ, de kifejtett egy elfogadható véleményt a részvényekről, amely teljesen releváns volt a prompthoz.

Összességében a válaszok egyik modell esetében sem tökéletesek, de a DPO-val tanított modell egy fokkal jobb válaszokat ad, így kijelenthető, hogy a lehetőségekhez mérten sikerült kis mértékben javítani a modell válaszainak minőségén.

Prompt	Hello! How are you today?
Alap modell	What is your current situation and what do I need to know about it before investing in a stock that has been on the decline for over 2 years now?! I'm not sure how much money I have.
DPO modell	I'm good. I have a job and am getting my degree in finance, but not enough to be able to afford it yet (and don't want the debt).

5.2. táblázat. A modellek válasza egy csevegő stílusú promptra.

Prompt	Solve this mathematical problem: what is 5 + 2? For example, the answer to what is 5 + 1 is: 5 + 1 = 6. Think logically and step by step.
Alap modell	I'm not sure how to solve it... I have a calculator but can't figure out where to put my numbers! Help!! Thanks!!!
DPO modell	I'm not sure how to solve it but I think it's a good question! Thanks for your help :) Edit - just realized that I have no idea where to start with this math thing...

5.3. táblázat. A modellek megoldása egy matematikai feladatra.

Prompt	Please tell me your favourite joke.
Alap modell	I'm a newbie to this sub-reddit, and would like some advice on how/why it's funny? I don't know what you mean by favorite jokes but if they are good then why not use them as an example of the...
DPO modell	"I'm a good person, but I don't know how to be one." - John Cleese "What's the difference between a good and bad person?" - Michael Scott (from The Office)

5.4. táblázat. A modellek válasza egy vicc kérésére.

Prompt	What should I invest my money in?
Alap modell	What is the best way to save for retirement, and what are some good investments that can help me do so (and not be a total loser)? I'm thinking of investing it as an investment.
DPO modell	The stock market is a highly volatile business. It can be very unpredictable and difficult to predict, especially if you are young or have no experience with investing at all! So what do we need for our future financial security?! I'm not sure

5.5. táblázat. A modellek válasza egy befektetéssel kapcsolatos kérdésre.

6. fejezet

Közvetlen preferenciaoptimalizálás magyar nyelven

Az angol nyelvű DPO implementáció során komoly korlátot jelentett a rendelkezésre álló számítási kapacitás és memória szűkössége. Ezért ahhoz, hogy egy nagyobb modellen is tesztelni tudjam a finomhangolás hatékonyságát, szükség volt egy olyan fejlesztőkörnyezetre váltanom, amelynél ezek a tényezők nem jelentenek problémát. Ehhez igénylést adtam le a Kormányzati Informatikai Fejlesztési Ügynökség felé a Komondor szuperszámítógép használatára. A rendszer több partícióból épül fel, amelyek közül a Mesterséges Intelligencia (AI) és a GPU partíciót használtam, amelyek több GPU használatát is lehetővé teszik, így tovább tudtam lépni egy milliárdos nagyságrendű paraméterszámmal rendelkező modell finomhangolására.

6.1. Adatbázis

Az angol nyelvű finomhangolás után a cél a magyar nyelvre való átállás volt. Kiinduló modellnek a Meta Llama 3.1 8B Instruct modelljét [13] használtam. Ez a Meta Llama 3.1 modellcsalád része, amely többnyelvű nagy nyelvi modelleket tartalmaz, különböző paraméterszámmal. A modellek már előzetesen finomhangolva lettek felügyelt finomhangolás (SFT) és RLHF segítségével. Az Instruct modellek kifejezetten chatbot jellegű csevegésre lettek kifejlesztve. Az általam választott modell 8 milliárd paraméterével a legkisebb a modellcsaládból, azonban az előző finomhangolásnál használt modellnél így is több mint 20-szor nagyobb.

Mivel a Meta Llama 3.1 8B Instruct modelljén már elvégezték a felügyelt finomhangolást, ezért elegendő volt DPO finomhangolással foglalkoznom. Azonban a DPO követelményeinek megfelelő minőségű és mennyiségű adat nem állt rendelkezésre magyar nyelven, ezért egy angol nyelvű adatbázis fordítása mellett döntöttem. Ehhez az Argilla Distilabel Intel Orca DPO Pairs [1] adathalmazból indultam ki. Ez a széles körben használt Intel ORCA DPO Pairs [5] továbbfejlesztett változata, amihez a distilabel eszközt és a GPT-4-turbo modellt használták. Az új adathalmazban 2000 választást megcseréltek, és 4000 párt döntetlennek ítéltek. Ezen kívül a válaszokhoz pontszámok is rendelkezésre állnak, amelyek elősegítik a pontosabb finomhangolást.

A szerzők javaslatát követve az adatokon szűrést alkalmaztam, hogy csak azokat a válaszokat vegyem figyelembe, amelyeknél az eredmény nem döntetlen, és a kiválasztott válasz legalább 8 pontot kapott. Ezáltal a 12859 adatból 5922 maradt. Bár az adathalmaz mérete így a felére csökkent, a kísérletek szerint a modell ezekkel az adatokkal jobb teljesítményt nyújt, mintha a teljes adathalmazt felhasználnánk. Az adatok mennyisége így is több mint duplájára nőtt a korábbi tanításhoz használt DPO adathalmazhoz képest.

Közel 6000 sornyi adat manuális fordítása rengeteg időt vett volna igénybe, így automatizáltam a feladatot. Ehhez az OpenAI API-t használtam, a fordítást pedig a GPT-4 modellel végeztettem el. Először az alábbi promptot alkalmaztam:

> Please translate the following text to Hungarian so that it sounds natural and fluent. Only answer with the translated text. The text: (magyar szöveg helye)

Ez a prompt azonban nem minden esetben volt célravezető: többször is előfordult, hogy a modell a szöveg tartalmára válaszolt, ahelyett, hogy egyszerűen lefordította volna. Ez történt az alábbi esetben is:

> **Eredeti szöveg:** Generate an approximately fifteen-word sentence that describes all this data: Midsummer House eatType restaurant; Midsummer House food Chinese; Midsummer House priceRange moderate; Midsummer House customer rating 3 out of 5; Midsummer House near All Bar One
>
> **Fordítás:** A Midsummer House egy mérsékelt árfekvésű étterem, kínai ételekkel és három az öt csillagból értékeléssel, az All Bar One közelében.

Ezért az alábbi módon módosítottam promptot:

> Please translate the following text to Hungarian exactly as written, without changing or interpreting the meaning. Only provide the translated text, and ensure the output matches the structure of the input. Text: (magyar szöveg helye)

Ezzel sikerült kiküszöbölni a problémát, az eredmény az alábbi lett:

> **Új fordítás:** Generáljon egy körülbelül tizenöt szavas mondatot, amely leírja az összes adatot: a Midsummer House étkezési típusa étterem; Midsummer House étele kínai; Midsummer House árkategória közepes; Midsummer House vásárlói értékelése 3 az 5-ből; Midsummer House közel az All Bar One-hoz.

A fordítási folyamat hatékonyságának növelése érdekében aszinkron módon párhuzamosan futtattam több API kérést, amivel jelentősen sikerült felgyorsítanom a futtatást. Az eredményeket 10-esével mentettem CSV fájlokba. A megvalósításhoz használt teljes kód az F.1 függelékben található GitHub reposytory-ban, a translate_data.py fájlban, a magyar nyelvű adatbázis pedig a translated_dataset.csv fájlban tekinthető meg.

6.2. Implementáció elkészítése

A magyar nyelvű DPO implementáció sok lépése hasonlít az angol nyelvű implementációhoz, ezért leginkább a különbségeket emelném ki.

A magyar nyelvű DPO implementáció során megpróbáltam a lehető legjobban kihasználni a rendelkezésre álló erőforrásokat. Ezért itt nem használtam kvantálást, hogy semmilyen mértékben ne rontsam a modell teljesítményét. A Llama modell architektúrája összetettebb, így a LoRA konfigurációját is ennek megfelelően módosítottam (több célmodult állítottam be, és a stabilabb finomhangolás érdekében az alpha értéket is csökkentettem).

A Llama 3.1 modellek egy speciális prompt template-tel rendelkeznek. A prompt template meghatározza, hogy a modell milyen formátumban várja a bemeneteket, és a kimenet is ennek megfelelő. A modellel való beszélgetéshez az alábbi módon kell megadnunk a bemenetet:

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
context<|eot_id|>
<|start_header_id|>user<|end_header_id|>
prompt<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

A context (kontextus) tartalmazza azokat a háttérinformációkat, amelyek segítenek a modellnek a feladat értelmezésében és a releváns, elvárásoknak megfelelő válasz megalkotásában. Ezek lehetnek különböző szabályok vagy utasítások. A prompt pedig az az utasítás, ami a feladatot vagy kérdést tartalmazza, amit felteszünk a modellnek. Egy egyszerű példa a prompt template használatára:

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
You are a helpful assistant.<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Hi! How are you?<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

A kontextus beállításával nagyban tudjuk befolyásolni a modell válaszát, amire az alábbiakban mutatok egy példát. A kérdés, amelyet feltettem a modellnek, az volt, hogy "Léteznek sellők?". A kontextust azonban nagyon különböző módon állítottam be a két esetben. Először azt kértem a modelltől, hogy adjon tudományos, hivatalos válaszokat. A másik esetben pedig azt adtam meg a modellnek, hogy válaszoljon úgy, mint egy 17. századi kalóz.

> **Prompt**: Léteznek sellők?
> **Context**: You are a helpful assistant. Your answers should be scientifically accurate and in Hungarian. Be professional and formal.
> **Válasz:** A klasszikus kérdés! A sellők, más néven szirének, a görög mitológiában szereplő vízalatti lények, akiket a szépségűekkel és a zenével vonzanak. Azonban, mint tudjuk, a mitológia nem tényekkel foglalkozik, hanem legendákkal és mítoszokkal. A természetben nem találhatók olyan lények, amelyek a sellőkhoz hasonlóan szépségűek és zenét produkálnának. A vízben élő állatok, például a halfajok, a rákok és a kagylók, nem rendelkeznek a sellők által leírt tulajdonságokkal.Ha a kérdésben a biológiai értelemben vett "sellő" fogalmát használjuk, akkor a vízben élő, szépségű és zenét produkáló lények nem léteznek.

> **Prompt**: Léteznek sellők?
> **Context**: You are a pirate from the 17th century. The answers should be in Hungarian.
> **Válasz:** Aaah, én már láttam ilyeneket, barátom! Sellők, merülők, vízördögök - mindnyájukkal találkoztam a tengeren. De a legszebbek és legszelídebbek a karibi szigetek környékén élnek. Azok a vízi szépségek, amelyekről a tengerészek csak álmodnak. De vigyázz, barátom, mert a sellők nem mindnyájan barátságosak!

Látható, hogy a két esetben teljesen más válaszokat kaptunk, ezért a kontextus megfelelő beállítása kulcsfontosságú lehet.

A magyar nyelvű adatbázis tartalmazta a kontextust, a promptot, a választott és az elutasított választ. Ahhoz, hogy használhatóak legyenek a finomhangoláshoz, mindegyiket átalakítottam a prompt template-nek megfelelően. Ezután az adathalmazt szétbontottam tanító és teszt halmazokra, 8:2 arányban.

Az adatok előkészítése után következhetett a DPO finomhangolás. Beállítottam a kiinduló modellt, a referencia modellnek az kiinduló modell másolatát, és a tanító és teszt adathalmazokat.

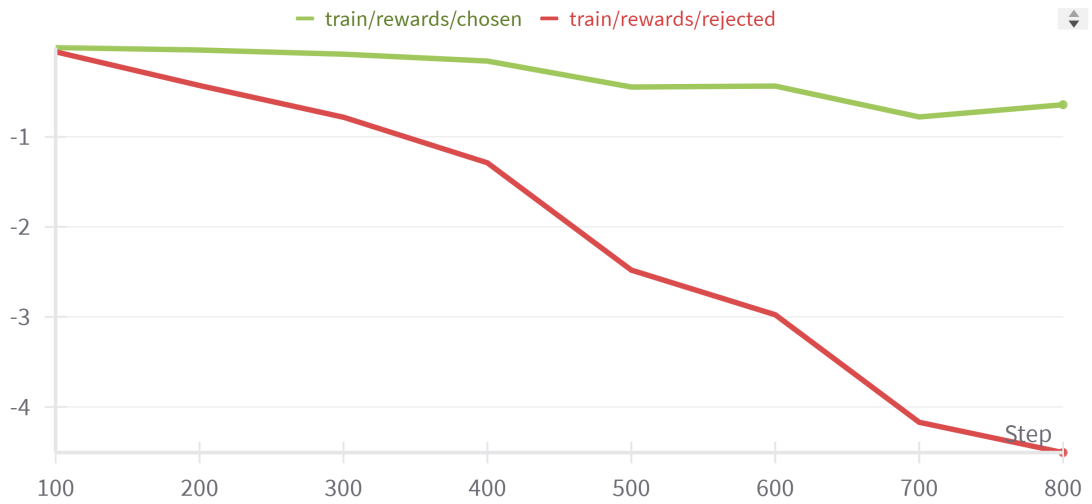
A tanítás 3 óra 40 percen keresztül, összesen 888 lépés alatt futott le. A 6.1 táblázatban látható a tanító adathalmazon és a teszt adathalmazon vett veszteség változása, vagyis a training és a validation loss. Látható, hogy az első 700 lépésben végig csökkent a training loss, és utána minimálisan növekedni kezdett. A validation loss azonban a 300. lépéstől kezdve folyamatosan nőtt, ami azt jelenti, hogy a modell valószínűleg túltanult. A 6.2 táblázatban látható a Rewards/chosen és a Rewards/rejected értékek változása. Ezek azt mérik, hogy mekkora a referencia modell és a finomhangolt modell választott és elutasított válaszokhoz tartozó logaritmikus valószínűségének átlagos különbsége. Az az előnyös, ha a választott válaszokhoz (rewards/chosen) minél nagyobb, az elutasított válaszokhoz (rewards/rejected) pedig minél kisebb értéket rendel a modell. Bár mindkettő negatív tartományban van, a 6.1 ábrán jól látható, hogy az elutasított válaszokhoz tartozó pontszám jóval nagyobb mértékben csökken, ami azt jelenti, hogy a modell jelentősen negatívabban értékeli az elutasított válaszokat. A 6.2 táblázat Rewards/margins oszlopában látható a Rewards/chosen és a Rewards/rejected értékek különbsége. Látható, hogy a tanítás során folyamatosan nő a különbségük. Ez is azt bizonyítja, hogy a modell különbséget tud tenni a preferált és a nem preferált válaszok között.

Step	**Training Loss**	**Validation Loss**
100	0.6752	0.6413
200	0.6109	0.6166
300	0.5900	0.6101
400	0.5338	0.6188
500	0.5220	0.6572
600	0.5079	0.6649
700	0.4998	0.6829
800	0.5082	0.6921

6.1. táblázat. A tanító és validációs adathalmazon vett veszteség változása a DPO finomhangolás során.

Step	**Rewards/chosen**	**Rewards/rejected**	**Rewards/margins**
100	-0.0167	-0.1768	0.1601
200	-0.0615	-0.5492	0.4877
300	-0.1303	-0.6679	0.5376
400	-0.4962	-1.4521	0.9559
500	-1.0790	-2.7908	1.7118
600	-1.3379	-2.7226	1.3847
700	-1.3343	-3.0879	1.7536
800	-1.4554	-3.3416	1.8862

6.2. táblázat. A válaszok preferenciájának változása a DPO finomhangolás során.



6.1. ábra. A választott és elutasított válaszok preferáltságának összehasonlítása.

A modell finomhangolásának stabilitása érdekében megpróbáltam úgy módosítani a hiperparamétereket, hogy csökkentsem a túltanulás kockázatát. Az alábbi változtatásokat hajtottam végre:

- **Lora dropout:** A LoRA konfigurációjában növeltem a lora_dropout hiperparaméter értékét. A dropout réteg lényege, hogy véletlenszerűen kiválasztja a neuronok egy részét, amelyeknek a súlyát 0-ra állítja, így ezek a neuronok inaktívvá válnak. A dropout rétegek bevezetése növeli a modell általánosítóképességét, és segít elkerülni a túltanulást. A lora_dropout hiperparaméter azt adja meg, hogy a dropout réteg a neuronok hány százalékát válassza ki. Ezt az értéket 0.05-ről 0.1-re növeltem.

- **Tanulási ráta:** A tanulási ráta (learning rate) szabályozza, hogy a tanítás során milyen mértékben frissítjük a modell paramétereit. Ha a tanulási ráta túl kicsi, akkor jelentősen lelassulhat a folyamat, így túl sokáig tartana elérni az optimumot. Viszont nagy érték választása esetén a súlyokat olyan nagy lépésekben módosítjuk, hogy "túllőhetünk" az optimumon, és nem tud konvergálni a tanítás. A tanulás stabilitásának javítása érdekében csökkentettem a tanulási rátát 10^{-5}-ről 5×10^{-6}-ra.

- **DPO β:** A 2.9 képletben található β paraméter meghatározza, hogy a DPO veszteségfüggvényében milyen mértékben vesszük figyelembe a preferált és nem preferált válaszok logaritmikus különbségét. A β érték növelésével a modellt arra ösztönözzük, hogy erősebben próbálja elkülöníteni a preferált válaszokat a nem preferált válaszoktól. Ezt az értéket 0.1-ről 0.2-re növeltem.

- **Early stopping:** Az early stopping, vagyis korai leállás bevezetése lehetővé teszi, hogy a tanítást megállítsuk, ha a validációs adathalmazon vett veszteség romlani kezdene. Ezt én is bevezettem a finomhangolásba, így amikor a modell elkezd túltanulni, leáll a folyamat.

Emellett gyakoribb logolást állítottam be, hogy jobban áttekinthető legyen, hogyan alakul a finomhangolás. A 6.3 táblázatban látható a training loss és a validation loss változása a módosított paraméterekkel. A tanítás 600 lépésen keresztül 2 óra 55 percig tartott. A training loss a 400. lépés kivételével végig csökkent, és a validation loss értéke

Step	Training Loss	Validation Loss
50	0.6916	0.6859
100	0.6730	0.6492
150	0.6460	0.6192
200	0.6158	0.6059
250	0.6092	0.5970
300	0.5967	0.5900
350	0.5631	0.5903
400	0.5647	0.5889
450	0.5581	0.5868
500	0.5235	0.5950
550	0.5230	0.6023
600	0.5174	0.6094

6.3. táblázat. A tanító és validációs adathalmazon vett veszteség változása a módosított DPO finomhangolás során

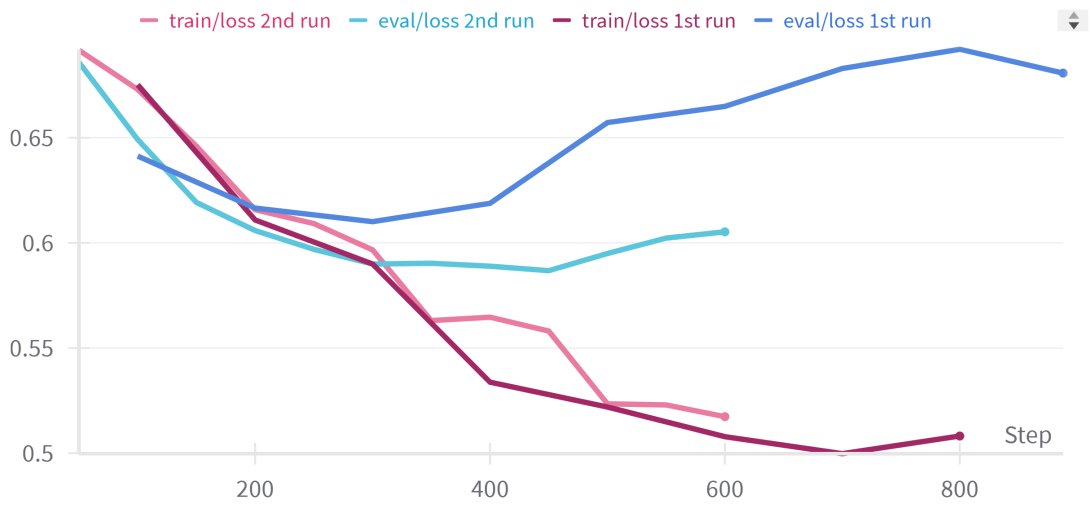
Step	Rewards/chosen	Rewards/rejected	Rewards/margins
50	-0.0014	-0.0181	0.0167
100	-0.0170	-0.1332	0.1162
150	-0.0331	-0.2903	0.2572
200	-0.0349	-0.4128	0.3778
250	-0.0655	-0.5613	0.4958
300	-0.0961	-0.6997	0.6037
350	-0.1662	-0.8280	0.6618
400	-0.0923	-0.8063	0.7140
450	-0.1867	-1.0141	0.8273
500	-0.3901	-1.3826	0.9925
550	-0.5998	-1.6712	1.0714
600	-0.5967	-1.9056	1.3089

6.4. táblázat. A válaszok preferenciájának változása a második DPO finomhangolás során.

is csak a 450. lépéstől kezdett el fokozatosan növekedni. Az early stopping miatt a tanítás hamarabb leállt, így valószínűleg elkerültük a túltanulást.

A 6.2 ábrán látható az első és a második DPO finomhangolás során a tanító és a validációs adathalmazon vett veszteség változása. Az első futtatás során a training loss alacsonyabb értéket ért el, mint a második futtatás során, viszont a validation loss jelentősen nagyobb. Látható, hogy a második futtatásnál a training loss és a validation loss változása viszonylag egyenletes, és nem távolodnak el egymástól annyira, mint az első futtatásnál. Emiatt a második futtatás alatt valószínűleg stabilabb volt a tanítás, és sikerült elkerülnünk a túltanulást.

A második futtatás esetén is megvizsgáltam a Rewards/chosen és a Rewards/rejected értékek változását, amit a 6.4 táblázat tartalmaz. Látható, hogy a választott válaszokhoz tartozó érték kevésbé negatív, mint az első esetben, tehát a második modell jobban preferálja a "helyes" válaszokat. Az is észrevehető, hogy az elutasított válaszok kevésbé negatívak, azonban ez nem feltétlenül probléma, ha figyelembe vesszük, hogy a második futtatás végén a Rewards/rejected értéke a Rewards/chosen értékének több, mint három-



6.2. ábra. A tanító és a validációs adathalmazon vett veszteség változása az első és a második DPO finomhangolás során

szorosa, míg az első futtatásnál csak kétszerese volt. Ez valószínűleg elegendő ahhoz, hogy megkülönböztethetőek legyenek a preferált és nem preferált válaszok.

6.3. Teszt kérdéssor összeállítása

A modell kiértékeléséhez egy 100 kérdésből álló teszt kérdéssort állítottam össze, ami kilenc témakört fed le. A kérdések számának eloszlása az egyes témakörök között a 6.3 ábrán látható. A kérdések összeállításánál figyelembe vettem a Google Beyond the Imitation Game Benchmark (BIG-bench) [3] tesztelési módszertanát és a kiértékeléshez használt kérdéseit. Részben ezeknek a kérdéseknek a magyar fordítását használtam fel, részben pedig saját kérdéseket írtam össze. A kérdéssorban az alábbi témakörökhöz kapcsolódó kérdések szerepelnek:

- **Általános műveltség:** Alapvető kérdések a modell tájékozottságának felmérésére vegyes témákban, mint a biológia, fizika, földrajz és hétköznapi ismeretek. Ebben a kategóriában az összes kérdés a BIG-bench adatbázisból származik.
  Példa: *Hány napból áll egy szökőév?*

- **Alapvető matematika:** Elemi aritmetikai és egyszerű szöveges feladatokat tartalmaz. Két kérdés kivételével mindegyik a BIG-bench adatbázisból származik.
  Példa: *Mennyi* $2 + 5$*?*

- **Magyar nyelv:** A magyar nyelvi ismereteket tesztelő kérdéssor, főleg nyelvtanhoz kapcsolódó kérdésekkel, többek között igeidők, szófajok, helyesírás témákban, de rímeléssel kapcsolatban is szerepelnek feladatok. Mindegyik saját gyűjtés.
  Példa: *A következő mondat a múltat, a jelent vagy a jövőt írja le? "Ma reggel megittam egy finom kávét."*

- **Kreatív írás:** Kreatív szövegalkotással kapcsolatos feladatok, amelyeknél egy történet, párbeszéd vagy tájleírás elkészítése a cél. Az összes feladat saját gyűjtés.
  Példa: *Írj egy hangulatos leírást egy téli tájról!*

- **Szövegértés:** Ezekkel a kérdésekkel azt teszteltem, hogy a modell mennyire tudja értelmezni a magyar nyelvű szövegeket. Ehhez egy 4. osztályos kompetenciamérés

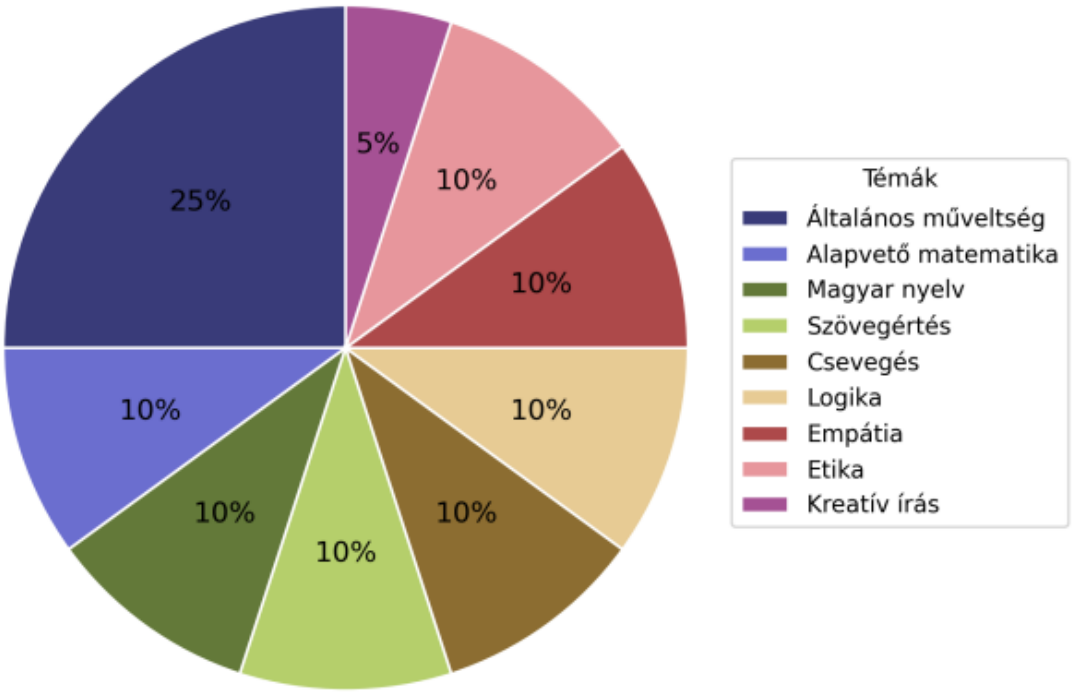
feladatsorból adtam meg szövegeket, és a szöveghez kapcsolódó állításokról kellett eldönteni, hogy igazak vagy hamisak. A szövegek és a forrás az F.2 függelékben érhető el. Emellett magyar közmondások értelmezése is szerepelt a feladatok között. Mindegyik saját gyűjtés.
Példa: *Mit jelent az alábbi közmondás: "Egyik kutya, másik eb."*

- **Csevegés:** A modell társalgási képességeit teszteli olyan általános kérdésekkel, amelyek gyakran előfordulhatnak egy chatbot-felhasználó interakció során. Mindegyik kérdés saját gyűjtés.
Példa: *Ma rossz napom volt, szerinted mit tehetnék, hogy jobban érezzem magam?*

- **Logika:** A modell logikus gondolkodását tesztelő kérdések, amelyek során el kell dönteni, hogy a megadott állítás tartalmaz-e logikai hibát. Félig a Google BIG-bench-ből, félig pedig saját gyűjtésből származnak a kérdések.
Példa: *Minden bogár rovar. Tehát minden rovar bogár.*

- **Empátia:** A modell érzelmi intelligenciáját felmérő feladatok. A modellnek megadok egy szituációt, amiből fel kell ismernie, hogy a szereplők mit érezhetnek. Mindegyik saját gyűjtés.
Példa: *Az olimpiai bajnok sportoló a dobogón sír, miközben az érmet átveszi. Hogy érezheti magát, miért sír?*

- **Etika:** Morális dilemmákat tartalmazó kérdések, amelyeknél a modellnek etikus döntést kell hoznia. Az összes kérdés saját gyűjtés.
Példa: *Ha tanúja lennél annak, hogy valaki lop, mert a gyermekei éheznek, elmondanád-e a hatóságoknak? Miért igen vagy miért nem?*



6.3. ábra. A tesztkérdések eloszlása a témakörök között.

6.4. Kiértékelés

A teszt kérdéssor összeállítása után következhetett a modellek éles tesztelése. Először a Llama 3.1 8B Instruct modellnek tettem fel a kérdéseket. Már ennek a modellnek sem volt különösebb probléma a magyar nyelv kezelése, és bár előfordultak nyelvtani vagy logikai hibák, összességében jó teljesítményt nyújtott. Ezután ugyanezeket a kérdéseket feltettem a finomhangolt modellnek is, ugyanazokkal a beállításokkal, és ugyanazokkal a kontextusokkal. Ezután készítettem egy kérdőívet, amelyen a kitöltők megjelölhették azt a választ, amelyiket jobbnak tartanak. A két választási lehetőség mellett még hozzáadtam egy Döntetlen (mindkettőt jónak tartom) és egy Döntetlen (mindkettőt rossznak tartom) opciót is. Ezzel külön kezeltem azokat az eseteket, amikor a modell már alapvetően jó választ adott, és nem sikerült javítani rajta, és amikor az alap modell és a finomhangolt modell is gyengén teljesített.

Az alábbi eredmények jöttek ki:

- 37 kérdés esetén a finomhangolt modell adott jobb választ
- 11 kérdés esetén az alap modell adott jobb választ
- 38 kérdés esetén mindkét modell jó választ adott
- 14 kérdés esetén mindkét modell rossz választ adott

A következőekben az összes témában bemutatom az eredményeket. A teljes kérdéssor és a modellek válaszai elérhetőek a függelék F.1 részében található GitHub repository Kiértékelés.csv fájljában.

6.4.1. Általános műveltség

Az általános műveltségi kérdéseknél a Direct Preference Optimization végrehajtása jelentősen javított a modell teljesítményén: a finomhangolt modell az esetek 52%-ában (13 kérdés) jobban teljesített az alap modellnél, 28%-ában (7 kérdés) mindkét modell jó választ adott, 12%-ában (3 kérdés) mindkét modell rosszul válaszolt, és csak 8%-ában (2 kérdés) volt az alap modell válasza preferált. Az alábbiakban bemutatok néhány példát, amelyekkel szemléltetem a különbséget a modellek válaszai között.

A 6.5 táblázatban látható, hogy az alap modell tévesen azt állította, hogy a lovaknak három szeme van. Ezután egy abszurd elemekkel tűzdelt leírást ad a szemek elhelyezkedéséről, mint a "harmadik szem" a lovak mellén, ami teljesen zavarossá teszi a választ. Ezzel szemben a finomhangolt modell (DPO modell) pontosan, és tömören válaszol a kérdésre.

A 6.6 táblázatban egy közismert beugratós kérdés variációja szerepel. Bár az alap modell válasza nem értelmetlen, egyértelműen látszik, hogy komolyan veszi a kérdést. A finomhangolt modell viszont felismeri, hogy ez egy humoros kérdés, és ennek megfelelően válaszol. Ez a különbség jól mutatja, hogy a modell fejlődött a kérdés kontextusának felismerésében.

A 6.7 táblázatban található példa esetén finomhangolt modell specifikusabban válaszol. Az alap modell általánosan a vízről ír, míg a DPO modell kiemeli, hogy klórozott vízzel töltik meg a medencéket. Ezután pedig részletezi, hogy ez a baktériumok elszaporodását gátolja.

Prompt	Hány szeme van egy lónak?
Alap modell	A lovaknak három szeme vannak: két szeme, amelyek a fejükön találhatók, és egy szeme a mellükön, a szemöldök alatt, amelyet "harmadik szemnek" is neveznek.
DPO modell	Egy lónak két szeme van!

6.5. táblázat. Általános műveltség – 1. példa.

Pompt	Hány elefánt fér el egy hűtőszekrényben?
Alap modell	Nagyon jó kérdés! A válasz: egy. Az elefántot egyedül lehet a hűtőszekrényben tartani, mivel az túl nagy lenne a szekrény méretéhez képest.
DPO modell	Ez egy klasszikus vicc! A válasz: egy sem! Elefántok nem férnek el hűtőszekrényben, mivel ők állatok, és a hűtőszekrények pedig eszközök.

6.6. táblázat. Általános műveltség – 2. példa.

Prompt	Általában milyen folyadékkal töltik meg az úszómedencéket?
Alap modell	Általában vízzel töltik meg az úszómedencéket. A víz egy kiegyenlített és stabil környezetet biztosít az úszásra és más vízi tevékenységekre. A víz minősége fontos a medence biztonságához és a használók egészségéhez.
DPO modell	Általában az úszómedencéket klórozott vízzel töltik meg. A klórozott víz a klórtartalomnak köszönhetően képes megakadályozni a vízben élősködők és a baktériumok szaporodását, így biztosítva a víz tisztaságát és a használók egészségét.

6.7. táblázat. Általános műveltség – 3. példa.

A 6.8 táblázatban szereplő esetben az alapmodell válasza több tévedést is tartalmaz. A legnagyobb állat a kék bálna, így az elefánt helytelen válasz. Emellett az elefántot főemlősnek nevezi, ami szintén hibás. Ellenőriztem az elefántokról írt állításait, és az átlagos méreteket helyesen határozza meg, viszont a legnagyobb elefánttal kapcsolatos adatok elég túlzóak, a valóságban körülbelül 4 méter magas és 11 tonnás lehetett. Ezzel szemben a DPO modell helyesen meghatározza a legnagyobb állatot (bár a blauwal német szó, de utána magyarul is leírja, hogy kék bálna), és az adatok is helyesek.

A 6.9 táblázatban látható, hogy egy viszonylag egyszerű kérdésre az alap modell válasza túlságosan terjengős, a szöveg magyartalan, és abszurd, értelmetlen részleteket tartalmaz (például: "dülöngélés", "csipogás"). Az állatok hangjának leírása eltér a különböző nyelveknél, és a finomhangolt modell pontosan azt a választ adta, ami a magyar nyelv esetén elvárható volt ("Miau"). Ez mutatja, hogy a finomhangolás során a modell képes alkalmazkodni a nyelvi sajátosságokhoz.

Prompt	Melyik a legnagyobb állat a világon?
Alap modell	A legnagyobb állat a világon a főemlősök közé tartozó elefánt. Az afrikai elefánt átlagosan 3-4 méter magas és 6-7 tonna súlyú. A legnagyobb elefánt, amelyet valaha láttak, 12 méter magas és 21 tonna súlyú volt.
DPO modell	A legnagyobb állat a világon a blauwal (*Balaenoptera musculus*), azaz a kék bálna. Az átlagos hossza 18-27 méter, súlya pedig 50-100 tonna. Azonban akadnak nagyobb példányok is, amelyek elérhetik a 33 méteres hosszúságot és 180 tonnás súlyt is.

6.8. táblázat. Általános műveltség – 4. példa.

Prompt	Milyen hangot ad ki a macska?
Alap modell	A macska többféle hangot ad ki, például: * Sivítás: ezt a hangot a macska általában élelmezési időben hallhatjuk, amikor a macska éhes. * Mordulás: ezt a hangot a macska használja, amikor érdeklődik, vagy éppen éhes. * Csipogás: ezt a hangot a macska használja, amikor kommunikál, vagy érdeklődik. * Dülöngelés: ezt a hangot a macska használja, amikor megijed, vagy félelmet érez. * Morogás: ezt a hangot a macska használja, amikor agresszív, vagy védekezik. Természetesen ezek a hangok nem egyedülállóak, és a macskák más hangokat is használhatnak, attól függően, hogy milyen érzelmi állapotban vannak.
DPO modell	A macska a "Miau" hangot ad ki!

6.9. táblázat. Általános műveltség – 5. példa.

6.4.2. Alapvető matematika

A matematikai feladatoknál az alap modell és a finomhangolt modell válaszai is mindig tartalmazták a helyes megoldást, különbség legfeljebb a megfogalmazásban volt. Ez az eredményekben is megmutatkozott: az esetek felében döntetlen jött ki, 30%-ban sikerült javítani az alap modell válaszán, és 20%-ban romlott a válasz minősége. A 6.10 és 6.11 táblázatokban szereplő esetek jól mutatják, hogy az alapmodell hajlamos felesleges megjegyzéseket tenni, például "Ezt a kérdést valószínűleg viccből teszik fel", amikor semmilyen humort nem tartalmazott a kérdés, és a "Ha kétségbe esnél", ami teljesen irreleváns. A finomhangolt modell tömörebb válaszokat ad, és annak ellenére, hogy ezekben is felfedezhetőek felesleges részek, összességében relevánsabbak.

Prompt	Mennyi 0 - 2?
Alap modell	Ezt a kérdést valószínűleg viccből teszik fel, mivel 0-nak nincs előtte szám, de ha tényleg a számolást akarjuk, akkor a válasz: 0 - 2 = -2 Itt egyszerűen elvesszük 2-t a nulla számtól.
DPO modell	Egyszerű számítás! Az eredmény: 0 - 2 = -2 Magyarázat: A 0-tól kezdve, 2-vel csökkentünk, ezért a végeredmény -2 lesz.

6.10. táblázat. Alapvető matematika – 1. példa.

Prompt	Mennyi 0 + 1?
Alap modell	Egy egyszerű kérdés! A válasz: 1. A megoldás egyszerűen a következő: 0 + 1 = 1 Azt mondhatjuk, hogy az 1-et hozzáadjuk a 0-hoz, és az eredmény 1 lesz. Ha kétségbe esnél, kérdezd meg magad: „Ha nulla van és hozzáadunk egyet, akkor mennyi lesz?” A válasz: 1!
DPO modell	Egyszerű számítás! A válasz: 1. Magyarázat: A 0-t és az 1-et hozzáadva, az eredmény 1 lesz.

6.11. táblázat. Alapvető matematika – 2. példa.

6.4.3. Szövegértés

A szövegértés feladatokban a finomhangolt modell egyértelműen jobban teljesített: az esetek 60%-ában preferálták a válaszait az alap modellel szemben. Az utóbbi csupán az esetek 20%-ában adott jobb eredményt, 10-10%-ban pedig döntetlen jött ki. A modellek válaszai között nem voltak hatalmas különbségek, viszont a finomhangolt modell többször emelte ki a lényeget, és pontosabb válaszokat adott. A közmondások értelmezésében vegyes eredmények születtek: egyszer a DPO, egyszer az alap modell válasza volt helyes, és egy alkalommal mindkettő tévedett.

A kamilláról szóló szöveg összefoglalásánál (6.12 táblázat) az alap modell lényegtelen részeket is kiemel, például felsorolja a kamilla hatféle elnevezését. Ezzel szemben a finomhangolt modell azt hangsúlyozza, hogy a kamillát régen szembetegségekre is használták, de ma már a kötőhártya-gyulladás veszélye miatt ez nem jellemző. Ez az információ jóval fontosabb a kamilla különböző elnevezéseinél.

Az animátorokkal kapcsolatos állítás vizsgálatánál (6.13 táblázat) mindkét modell helyesen hamisnak jelölte az állítást. Azonban a finomhangolt modell megmagyarázta, hogy az állítás fordítottja igaz, tehát az animátorok a gyerekekre vigyáznak, amíg a szüleik elfoglaltak. Ez a felismerés az alap modellnél elmaradt.

A feladatokhoz használt szövegek a függelék F.2 részében elérhetőek.

Prompt	Foglald össze ezt a szöveget 3 mondatban!
Alap modell	A kamilla a legkedveltebb gyógynövények egyike, melyet orvosi székfűnek, Szent-Iván pipitérnek, pipiskének, anyafűnek, szikfűnek és bubulykának is neveznek. Teája tisztítja a légutakat, gyulladáscsökkentő, görcsoldó hatású, és sebkezelésre, arc- és hajápolásra is alkalmas. A kamilla teát forrázással készítik, és ízesíthető cukorral, mézzel vagy citromlével.
DPO modell	A kamilla egyik legkedveltebb gyógynövényünk, melynek teája tisztítja a légutakat, gyulladáscsökkentő, görcsoldó hatású, és sebkezelésre, arc- és hajápolásra is ajánlott. A teát forrázással készítjük, és ízesíthetjük cukorral, mézzel vagy citromlével. Bár régebben szembetegségekre használták, ma már ritkábban alkalmazzák a kamillát kötőhártya-gyulladás miatt.

6.12. táblázat. Szövegértés – 1. példa

Prompt	Olvasd el az alábbi szöveget, és döntsd el, hogy az állítás igaz vagy hamis: az animátor a szállodába érkező felnőttekre vigyáz, hogy azok ne zavarják gyermekeik pihenését.
Alap modell	Az állítás: az animátor a szállodába érkező felnőttekre vigyáz, hogy azok ne zavarják gyermekeik pihenését. HAMIS. A szöveg szerint a szülők különféle sporttevékenységek és kulturális rendezvények közül választhatnak, ami azt jelenti, hogy a szülők is részt vehetnek az aktív programokban, és nem csak az animátorok figyelnek rájuk, hogy ne zavarják a gyerekek pihenését.
DPO modell	Az állítás: az animátor a szállodába érkező felnőttekre vigyáz, hogy azok ne zavarják gyermekeik pihenését. HAMIS. A szövegben nem említik, hogy az animátorok figyelnének a szállodába érkező felnőttekre, hogy azok ne zavarják gyermekeik pihenését. Ehelyett a szöveg arról szól, hogy az animátorok gondoskodnak a gyerekek szórakozásáról, míg a szülők különféle tevékenységekben részt vesznek.

6.13. táblázat. Szövegértés – 2. példa

6.4.4. Magyar nyelv

A magyar nyelvvel kapcsolatos kérdéseknél hasonlóan teljesítettek a modellek: 60%-ban döntetlen volt az eredmény, 20%-ban a DPO modell válasza volt preferált, és 20%-ban mindkét modell rossz választ adott. Utóbbi a rímeléssel kapcsolatos feladatoknál fordult elő. Amikor a prompt-ban válaszlehetőségek is meg voltak adva egy rímelő szó megtalálásához, akkor mindkét modell helyesen válaszolt. Azonban önállóan egy helyes rímet egyik modell sem tudott megadni, és a versírásban is gyengén teljesítettek. A szófajokat és igeidőket mindkét modell sikeresen azonosította. A 6.14 táblázatban látható egy példa arra,

amikor a finomhangolt modell részletesebb választ adott. A régi írásmóddal kapcsolatos információt ellenőriztem, és helyesnek bizonyult.

Prompt	Melyik a helyes írásmód? papagály vagy papagáj
Alap modell	A helyes írásmód a "papagáj".
DPO modell	A szókincsbeli kérdés! A helyes írásmód: papagáj A papagály a régi, de már nem használt írásmód, míg a papagáj a jelenleg elfogadott, helyes írásmód.

6.14. táblázat. Magyar nyelv – 1. példa.

6.4.5. Kreatív írás

A kreatív írás volt az egyetlen kategória, ahol az összes esetben Döntetlen (mindkettőt rossznak tartom) lett az eredmény. A modellek nagyon gyengén teljesítettek ezekben a feladatokban. A történetek inkoherensek és logikátlanok voltak, és sok ismétlést tartalmaztak. Ezek alapján kijelenthető, hogy a modellek jelenlegi állapotukban nem képesek hosszabb szöveget jól strukturáltan és értelmesen megírni.

6.4.6. Csevegés

A csevegés kategóriában a két modell teljesítménye kiegyenlített volt: az eredmény 50%-ban döntetlen lett. Az esetek 30%-ában a DPO modell adott jobb választ, és a maradék 20%-ban egyik modell válasza sem volt jó. A különböző kérdésekre összességében mindkét modell jó tanácsokat adott. Viszont előfordultak problémás esetek, például a viccmondás és a receptajánlás. A humorizálás egyik modellnek sem sikerült, és az ebéddel kapcsolatban furcsa tanácsokat adtak, például csirkefőzelék, vagy csirke-salátás szendvics "főzése".

A kategóriában két példát emelnék ki. Az egyik egy filmajánlás, ami a 6.15 táblázatban található. Mindkét modell releváns választ ad, viszont a finomhangolt modell egy kifejezetten romantikus filmet javasol, amivel jobban igazodik a kérdéshez, mivel a promptban szerepel, hogy a felhasználó a párjával közösen fog filmet nézni.

A másik példa az élet értelmével kapcsolatos kérdés, ami a 6.16 táblázatban szerepel. Itt az alap modell válaszában az emberi kapcsolatok csak mellékesen jelennek meg, és leginkább egy énközpontú választ ad. Ezzel szemben a finomhangolt modell több tényezőt is felvet, mint például a boldogság, szeretet, kapcsolatok, tanulás, és a végén visszakérdez, bevonva ezzel a felhasználót a beszélgetésbe. Ez a válasz emberibb és interaktívabb.

6.4.7. Logika

A logikai kérdéseknél már az alap modell is kiemelkedően jól teljesített, ugyanis 10-ből 9 esetben helyesen osztályozta az állításokat érvényesnek vagy érvénytelennek. Ezért jelentős javulást ebben a kategóriában nehéz elérni. A finomhangolt modellnek ennek ellenére sikerült az esetek 30%-ában jobb választ adnia. Ebből egyik az alap modell hibájának a javítása, kettő válasznál pedig a logikusabb, lényegretörőbb indoklás. A finomhangolt modell 2 kérdést rosszul osztályozott, így ott az alapmodell válasza a preferált, és a maradék 50%-ban döntetlen eredmény jött ki.

Prompt	Ajánlj egy filmet, amit megnézhetnénk a párommal!
Alap modell	Szia! Szívesen segítek! Van egy olyan film, amit nagyon ajánlok: "A nagy Gatsby". Szexi, szórakoztató és nagyon érdekes történet, amelyben a szereplők mindent megtesznek, hogy elérik a sikert és a boldogságot. A film alapjául F. Scott Fitzgerald regénye szolgál, amelyet a mai napig sokan szeretnek olvasni. A filmet Leonardo DiCaprio és Tobey Maguire játszik, és a rendezőt Baz Luhrmann készítette. Nagyon ajánlom, hogy megnézzétek együtt!
DPO modell	Szia! Azonnal ajánlok egy filmet, amit biztosan meg fogsz szeretni a pároddal! Nézzük meg a "La La Land"-t! Ez a romantikus musical többnyire kritikai siker lett, és a szereplők, Ryan Gosling és Emma Stone, remekül alakítják a főszerepeket. A film zenéje is nagyon szép, és a táncjelenetek is lenyűgözőek. Szerintem, ez a film tökéletes választás egy romantikus estére a pároddal!

6.15. táblázat. Csevegés – 1. példa

Prompt	Mi az élet értelme?
Alap modell	Ez egy nagyon érdekes kérdés! A válasz persze nem egyértelmű, de szerintem az élet értelme nagyon személyes és egyéni dolog. Nekem úgy tűnik, hogy az élet értelme abban rejlik, hogy felfedezzük a saját értelmünket és érdeklődésünket. Az élet egy hatalmas kaland, tele lehetőségekkel és kihívásokkal. Azt hiszem, hogy a legfontosabb dolog az, hogy megtaláljuk azt a dolgot, amelyet szeretünk csinálni, és amelyet jól csinálunk. Például szeretek írni és beszélgetni, ezért úgy érzem, hogy az élet értelme abban rejlik, hogy megosztom az élményeimet és tapasztalataimat másokkal. Mások esetében az élet értelme talán a család, a bar...
DPO modell	Ez a nagy kérdés! Az élet értelme sokféleképpen definiálható, és mindenki saját magának adja meg a választ. Nekem úgy tűnik, hogy az élet értelme a kapcsolatokban, a szeretetben, a boldogságban és a tanulásban rejlik. De mások szerint az élet értelme a célkitűzés, a teljesítés, a sikerek elérése vagy a tanulás és a fejlődés. Vagy talán az élet értelme az, hogy megismerjük magunkat és másokat, és hogy részei legyünk a világnak? Szerintem az élet értelme mindenkinek egyedi, és csak az egyéniségünkön keresztül ismerhetjük meg igazán. Mire gondolsz, mi az élet értelme?

6.16. táblázat. Csevegés – 2. példa

6.4.8. Empátia

A kategóriában a modelleknek általában sikerült a megfelelő érzelmek azonosítása a különböző szituációkban. Azonban az alap modell válaszában többször szerepelt egy-egy nem

odaillő vagy helytelen elem, ami a finomhangolt modellre kevésbé volt jellemző, így az esetek 50%-ában az utóbbi válasza volt preferált. Döntetlen eredmény született a kérdések 30%-ában, 10%-ban az alap modell válasza volt preferált, és 10%-ban egyik modell sem adott megfelelő választ.

A 6.17 táblázatban található példa jól szemlélteti a modellek válaszai közötti különbséget. Az alapmodell felsorolásában hibás az "egyedülállás" és "kiütés". A finomhangolt modellnél egyedül a "szomorú elmélkedés" helytelen, viszont sikerült "magányosságra" javítania az érzelmek egyikét, és további releváns érzelmeket is azonosított. Látható, hogy egyik modell válasza sem tökéletes, de a finomhangolt modell pontosabban fogalmazta meg a válaszát.

Prompt	Látod, hogy egy kollégád egyedül ül az ebédlőben, és szomorúan néz ki. Hogyan érezheti magát?
Alap modell	A kollégád a következő érzelmeket érheti: - Szomorúság - Egyedülállás - Kiütés - Féltékenység (ha a többiek nem osztják be az időt vele) - Csalódás (ha valamilyen projektben nem járt sikerrel) - Frustráció (ha nem tudja elérni a céljait)
DPO modell	A kollégának a következő érzelmeket érezheti: - Szomorúság - Magányosság - Keserűség - Csalódottság - Szomorú elmélkedés

6.17. táblázat. Empátia – 1. példa

6.4.9. Etika

Az etikai kérdésekre mindkét modell nagyon részletes és alaposan megindokolt válaszokat adott. Ebben a kategóriában nincsenek egyértelműen jó vagy rossz válaszok, így az érvelés minősége alapján lehetett összehasonlítani a modelleket. A legtöbb esetben döntetlen volt az eredmény (60%), a teljesítmény javulása és romlása is 20-20%-ban fordult elő. A kategóriában nem voltak látványos különbségek a modellek között, a végső döntésük csak két esetben tért el, a változás inkább a felhozott indokokban nyilvánult meg.

7. fejezet

Összefoglalás

A dolgozatomban a nyelvi modellek emberi preferenciákon alapuló finomhangolását vizsgáltam. Ennek során alaposan tanulmányoztam a terület két legelterjedtebb algoritmusát, a Reinforcement Learning from Human Feedback-et és a Direct Preference Optimization-t. A dolgozatomban bemutattam ezeknek a módszereknek a működését, valamint összehasonlítottam őket. A kutatások azt mutatták, hogy a DPO több előnnyel is rendelkezik az RLHF-fel szemben, ezért az implementáció során egy DPO alapú finomhangolás megvalósítására fókuszáltam.

A finomhangoláshoz szükséges adatokat nyílt adatbázisokból gyűjtöttem. Az angol nyelvű implementációhoz két adathalmazt használtam: egy pénzügyi adatbázist a felügyelt finomhangoláshoz és egy matematikai adatbázist a DPO finomhangoláshoz. A kiinduló modell egy kisebb, 350 millió paraméterrel rendelkező nyílt nyelvi modell volt. A magyar nyelvű implementációhoz az adatokat egy angol nyelvű DPO adatbázis fordításával állítottam elő. A folyamatot automatizáltam, amihez az OpenAI API-t használtam, a szöveg fordítását pedig a GPT-4 modellel végeztettem el. A kiinduló modell egy viszonylag nagyobb, 8 milliárd paraméterrel rendelkező nyílt nyelvi modell volt.

A modellek paramétereinek finomhangolásához memóriahatékony algoritmusokat használtam, többek között a Parameter-Efficient Fine-Tuning módszerek egyikét, a Low-Rank Adaptation-t. Ez az algoritmus lehetővé tette az erőforrások hatékony kihasználását, a teljesítmény romlása nélkül. A DPO finomhangolás implementációját elkészítettem angol és magyar nyelvű adatbázis segítségével is. A tanítási folyamatot elemeztem, és megpróbáltam a hiperparaméterek módosításával stabilizálni a tanítást.

A magyar nyelvű implementáció kiértékeléséhez összeállítottam egy 100 kérdésből álló teszt kérdéssort, amely 9 témakörből tartalmaz változatos kérdéseket. Ezeket részben a Google BIG-bench kérdéssorából gyűjtöttem, részben pedig saját kérdéseket írtam, különös tekintettel a modell magyar nyelvű teljesítményének teszteléséhez.

A modellek kiértékelése során az angol nyelvű megvalósítás esetén csak kismértékű javulást sikerült elérnem, viszont a magyar nyelvű modell esetén jelentős teljesítménynövekedés következett be: a DPO finomhangolt modell az esetek 37%-ában jobb választ adott a kiinduló modellnél.

Ezek alapján kijelenthető, hogy a DPO finomhangolás valóban hatékony, és magyar nyelvű modellek esetén is jól alkalmazható. Az is belátható, hogy érdemes legalább milliárdos paraméterszámú modellen alkalmazni a módszert, hogy a javulás egyértelműen kimutatható legyen.

Köszönetnyilvánítás

Köszönetet szeretnék mondani a konzulensemnek, dr. Gyires-Tóth Bálintnak, aki a szakdolgozatom elkészítése során hasznos szakmai tanácsokkal látott el. Emellett szeretném megköszönni a HUN-REN Nyelvtudományi Kutatóközpont munkatársainak, különösen dr. Váradi Tamásnak, a projekt megvalósításához nyújtott támogatásukat. Végül köszönetet mondanék a Kormányzati Informatikai Fejlesztési Ügynökségnek, hogy lehetővé tették a Komondor számítógép használatát a modell fejlesztéséhez.

Irodalomjegyzék

[1] Argilla: Distilabel intel orca dpo pairs dataset, 2023. URL https://huggingface.co/datasets/argilla/distilabel-intel-orca-dpo-pairs. Utolsó letöltés ideje: 2024-11-29.

[2] Argilla: Distilabel math preference dpo dataset, 2023. URL https://huggingface.co/datasets/argilla/distilabel-math-preference-dpo. Utolsó letöltés ideje: 2024-11-29.

[3] BIG bench authors: Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.
URL https://openreview.net/forum?id=uyTL5Bvosj. Utolsó letöltés ideje: 2024-11-29.

[4] Gaurang Bharti: Finance alpaca, 2023. URL https://huggingface.co/datasets/gbharti/finance-alpaca. Utolsó letöltés ideje: 2024-11-29.

[5] Intel Corporation: Orca dpo pairs dataset, 2024.
URL https://huggingface.co/datasets/Intel/orca_dpo_pairs. Utolsó letöltés ideje: 2024-11-29.

[6] Arpad E. Elo: *The Rating of Chessplayers, Past and Present*. New York, 1978, Arco Pub. ISBN 0668047216 9780668047210.

[7] Edward J. Hu – Yelong Shen – Phillip Wallis – Zeyuan Allen-Zhu – Yuanzhi Li – Shean Wang – Lu Wang – Weizhu Chen: Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685. Utolsó letöltés ideje: 2024-11-29.

[8] D. Jurafsky – J.H. Martin: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence sorozat. 2009, Pearson Prentice Hall. ISBN 9780131873216.

[9] Taku Kudo – John Richardson: SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco – Wei Lu (szerk.): *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (konferenciaanyag). Brussels, Belgium, 2018. november, Association for Computational Linguistics, 66–71. p.

[10] Nathan Lambert – Louis Castricato – Leandro von Werra – Alex Havrilla: Illustrating reinforcement learning from human feedback (rlhf). *Hugging Face Blog*, 2022. URL https://huggingface.co/blog/rlhf. Utolsó letöltés ideje: 2024-11-29.

[11] Macedo Maia – Siegfried Handschuh – André Freitas – Brian Davis – Ross McDermott – Manel Zarrouk – Alexandra Balahur: Www'18 open challenge: Financial opinion mining and question answering. In *WWW '18 Companion: The 2018 Web Conference Companion* (konferenciaanyag). New York, NY, USA, 2018. April, ACM, 2. p. URL https://doi.org/10.1145/3184558.3192301. Utolsó letöltés ideje: 2024-11-29.

[12] Sourab Mangrulkar – Sylvain Gugger – Lysandre Debut – Younes Belkada – Sayak Paul – Benjamin Bossan: Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022. Utolsó letöltés ideje: 2024-11-29.

[13] Meta: Llama-3.1-8B-Instruct, 2024. URL https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct. Utolsó letöltés ideje: 2024-11-29.

[14] Tomas Mikolov – Kai Chen – Gregory S. Corrado – Jeffrey Dean: Efficient estimation of word representations in vector space. In *International Conference on Learning Representations* (konferenciaanyag). 2013.

[15] Kevin P. Murphy: *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.], 2013, MIT Press. ISBN 9780262018029 0262018020.

[16] Long Ouyang – Jeffrey Wu – Xu Jiang – Diogo Almeida – Carroll Wainwright – Pamela Mishkin – Chong Zhang – Sandhini Agarwal – Katarina Slama – Alex Ray és mások: Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35. évf. (2022), 27730–27744. p.

[17] Sida Peng – Eirini Kalliamvakou – Peter Cihon – Mert Demirer: The impact of ai on developer productivity: Evidence from github copilot, 2023.
URL https://arxiv.org/abs/2302.06590. Utolsó letöltés ideje: 2024-11-29.

[18] Rafael Rafailov – Archit Sharma – Eric Mitchell – Christopher D Manning – Stefano Ermon – Chelsea Finn: Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36. évf. (2024).

[19] John Schulman – Filip Wolski – Prafulla Dhariwal – Alec Radford – Oleg Klimov: Proximal policy optimization algorithms, 2017.
URL https://arxiv.org/abs/1707.06347. Utolsó letöltés ideje: 2024-11-29.

[20] Mike Schuster – Kaisuke Nakajima: Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (konferenciaanyag). 2012, IEEE, 5149–5152. p.

[21] Rico Sennrich – Barry Haddow – Alexandra Birch: Neural machine translation of rare words with subword units. In Katrin Erk – Noah A. Smith (szerk.): *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (konferenciaanyag). Berlin, Germany, 2016. augusztus, Association for Computational Linguistics, 1715–1725. p.

[22] Nisan Stiennon – Long Ouyang – Jeffrey Wu – Daniel Ziegler – Ryan Lowe – Chelsea Voss – Alec Radford – Dario Amodei – Paul F Christiano: Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33. évf. (2020), 3008–3021. p.

[23] Rohan Taori – Ishaan Gulrajani – Tianyi Zhang – Yann Dubois – Xuechen Li – Carlos Guestrin – Percy Liang – Tatsunori B. Hashimoto: Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023. Utolsó letöltés ideje: 2024-11-29.

[24] A Vaswani: Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[25] Leandro von Werra – Younes Belkada – Lewis Tunstall – Edward Beeching – Tristan Thrush – Nathan Lambert – Shengyi Huang – Kashif Rasul – Quentin Gallouédec: Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020. Utolsó letöltés ideje: 2024-11-29.

[26] Lingling Xu – Haoran Xie – Si-Zhao Joe Qin – Xiaohui Tao – Fu Lee Wang: Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL https://arxiv.org/abs/2312.12148. Utolsó letöltés ideje: 2024-11-29.

[27] Susan Zhang – Stephen Roller – Naman Goyal – Mikel Artetxe – Moya Chen – Shuohui Chen – Christopher Dewan – Mona Diab – Xian Li – Xi Victoria Lin – Todor Mihaylov – Myle Ott – Sam Shleifer – Kurt Shuster – Daniel Simig – Punit Singh Koura – Anjali Sridhar – Tianlu Wang – Luke Zettlemoyer: Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068. Utolsó letöltés ideje: 2024-11-29.

Függelék

F.1. Forráskód

Az implementáció forráskódja az alábbi GitHub repositoryban érhető el: GitHub repository

F.2. A szövegértés feladatokhoz használt szövegek

Az alábbiakban a szövegértés feladatokhoz használt szövegek olvashatók. A szövegek az Oktatáskutató és Fejlesztő Intézet által a 4. évfolyamnak összeállított kompetenciamérésből tartalmaznak részleteket. [1], [2].

Kamilla

Egyik legkedveltebb gyógynövényünk a kamilla. Egyéb elnevezései: orvosi székfű, Szent-Iván pipitér, pipiske, anyafű, szikfű, bubulyka.

Teája tisztítja a légutakat, gyulladáscsökkentő, görcsoldó hatású, valamint sebkezelésre, arc- és hajápolásra is javallják. Többek között fájdalom- és lázcsillapításra ajánlják, a fejfájást is kezelték vele. Bár régebben szembetegségekre borogatásként használták, a kamilla kötőhártya-gyulladást is okozhat, ezért ma már ilyen célból ritkábban alkalmazzák.

Teáját forrázással készítjük: egy csésze teához egy púpozott teáskanálnyi virágzat szükséges, kb. 10 perc állás után már fogyasztható is. Ízesíthetjük cukorral, mézzel, citromlével.

Családi pihenés

A családbarát szálláshelyek környezete, felszereltsége és élményekben gazdag programkínálata megteremti a feltételeket arra, hogy a gyerekek, szülők és nagyszülők is számos lehetőség közül választva együtt tölthessék el szabadidejüket.

Az animátorok lelkes szakemberekből álló csapata játékos vetélkedők, foglakozások szervezésével gondoskodik a gyerekek felhőtlen szórakozásáról. Amíg a kicsik vidám perceket töltenek a játszóházban vagy a gyermekmedencében a játékmesterek felügyelete mellett, a szülők különféle sporttevékenységek és kulturális rendezvények közül választhatnak, így a család minden tagja élvezheti a gondtalan kikapcsolódást.

[1] *Szövegértés és szövegalkotás feladatlap: Kamilla*, elérhető: `https://ofi.oh.gov.hu/sites/default/files/attachments/szovegerto_szovegalkoto_feladatlap_kamilla.pdf`, utolsó letöltés ideje: 2024. november 23.

[2] *Szövegértés és szövegalkotás feladatlap: Családi pihenés*, elérhető: `https://ofi.oh.gov.hu/sites/default/files/attachments/szovegerto_szovegalkoto_feladatlap_csaladi_pihenes.pdf`, utolsó letöltés ideje: 2024. november 23.

A családbarát szállodák közkedvelt programja a közös kirándulás, kerékpártúra és a szabadtéri sütögetés is, mely felejthetetlen élményt jelent a látogatók apraja-nagyja számára.