Szegedi Tudományegyetem
Informatikai Intézet

Nyelvi modellekkel történő finomhangolás nélküli álhírfelismerés az AutoPrompt segítségével

Szakdolgozat

Készítette:
Sprok Dániel
programtervező
informatikus szakos
hallgató

Témavezető:
Dr. Berend Gábor
egyetemi docens

Szeged
2024

Feladatkiírás

A szakdolgozó feladata a véleményfelismerési feladaton bevezetett AutoPrompt módszer megismerése és tesztelése a nyelvi modellekkel történő álhírfelismerési feladat megoldása során. A vizsgálatok térjenek ki a módszer különböző előtanított nyelvi modellekkel történő használatára, illetve teszteljék le a módszer használhatóságát az eljárás hiperparamétereinek különböző megválasztása mellett. A szakdolgozat a nyelvi modellek osztályozási képességének finomhangolással elérhető eredményességének finomhangolás nélküli történő megközelítése a célja.

Tartalmi összefoglaló

- ***A téma megnevezése:***

Nyelvi modellekkel történő finomhangolás nélküli álhírfelismerés az AutoPrompt segítségével.

- ***A megadott feladat megfogalmazása:***

A feladatom annak meghatározása volt, hogy a véleményfelismerési feladaton bevezetett AutoPrompt módszer képes-e megközelíteni a nyelvi modellek osztályozási képességének finomhangolással elérhető teljesítményét egy álhírfelismerési feladat megoldása során. A vizsgálatok alatt különböző előtanított nyelvi modellek lettek alkalmazva, a módszer használhatósága pedig annak hiperparamétereinek különböző megválasztása mellett lett letesztelve.

- ***A megoldási mód:***

A feladat első felében a modellek finomhangolása kapott szerepet, a másodikban pedig az AutoPrompt módszer használatával elvégzett kísérletek. Minden finomhangolási kísérlet során öt epoch ment végbe, a kísérletek végeztével pedig a legjobban teljesítő modellek kerültek megtartásra. Az AutoPrompt programmal futtatott kísérletek alatt minden modell kilenc különböző vizsgálaton esett át annak érdekében, hogy a lehető legjobb eredményt érjem el. Ezek a kísérletek az alkalmazott paraméterek értékeiben különböznek.

- ***Alkalmazott eszközök, módszerek:***

A kísérletekhez a Google Colaboratory lett felhasználva mint virtuális futtatási környezet, a kódok a Python programozási nyelvben kerültek megírásra. A kísérletek során alkalmazott modellek betöltéséhez és azok használatához a HuggingFace nevezetű cég által kiadott és fejlesztett Transformers könyvtár került használatra. A vizsgálatokhoz elengedhetetlen, álhíreket és valós információkat tartalmazó adatkészlet felhasználáshoz, illetve a kísérletek eredményeinek megtekintéséhez a fent említett cég Datasets, illetve Evaluate könyvtára lett igénybe véve. A finomhangolási kísérletekhez a PyTorch nevezetű keretrendszer nyújtott segítséget. Az AutoPrompt módszer vizsgálatához természetesen az AutoPrompt program került futtatásra.

- ***Elért eredmények:***

A hagyományos finomhangolás meghaladja az AutoPrompt által elért eredményeket. Viszont azt érdemes észben tartani, hogy a több és nagyobb erőforrással rendelkező rendszereken az embernek több opciója és ideje van arra, hogy az AutoPrompt segítségével ennél jobban megközelítse vagy akár meghaladja a finomhangolás során elért eredményeket.

- ***Kulcsszavak:***

Mesterséges intelligencia, gépi tanulás, természetesnyelv-feldolgozás, nyelvi modellek, finomhangolás, AutoPrompt, álhírfelismerés, szövegosztályozás

Tartalomjegyzék

Feladatkiírás **2**

Tartalmi összefoglaló **3**

Tartalomjegyzék **5**

1. BEVEZETÉS **7**

2. FONTOSABB FELHASZNÁLT ESZKÖZÖK, KÖNYVTÁRAK BEMUTATÁSA **9**

2.1. GOOGLE COLABORATORY **9**

2.2. TRANSFORMERS KÖNYVTÁR **9**

2.3 PYTORCH, TORCH **10**

2.4 DATASETS KÖNYVTÁR **10**

2.5 EVALUATE KÖNYVTÁR **11**

2.6 AUTOPROMPT **11**

3. FELHASZNÁLT ADATKÉSZLET BEMUTATÁSA **12**

4. KÍSÉRLETEK ELVÉGZÉSÉNEK MÓDJA **13**

4.1. KÍSÉRLETEKHEZ FELHASZNÁLT MODELLEK **13**
4.1.1 BERT 13
4.1.2 RoBERTa 14
4.1.3 DeBERTa 15

4.2. FINOMHANGOLÁS MÓDSZER **15**

4.3. AUTOPROMPT MÓDSZER **16**

5. EREDMÉNYEK, KÖVETKEZTETÉSEK **19**

5.1. FINOMHANGOLÁS EREDMÉNYEK **19**
5.1.1. BERT-BASE-CASED 19
5.1.2 ROBERTA-BASE 19
5.1.3 DEBERTA-BASE 20
5.1.4 BERT-LARGE-CASED 21
5.1.5 ROBERTA-LARGE 22
5.1.6 DEBERTA-LARGE 23
5.1.7 FINOMHANGOLÁSI EREDMÉNYEK ÖSSZEHASONLÍTÁSA 24

5.2 AUTOPROMPT EREDMÉNYEK 24
5.2.1. BERT-BASE-CASED 24
5.2.2 ROBERTA-BASE 25
5.2.3 DEBERTA-BASE 26
5.2.4 BERT-LARGE-CASED 26
5.2.5 ROBERT-LARGE 27
5.2.6 DEBERTA-LARGE 28
5.2.7 AUTOPROMPT EREDMÉNYEK ÖSSZEHASONLÍTÁSA, LEGJOBB MODELL TOVÁBBI TESZTELÉSE 28

5.3 MÓDSZEREK ÖSSZEHASONLÍTÁSA 30

6. ÖSSZEFOGLALÓ 31

IRODALOMJEGYZÉK 32

NYILATKOZAT 34

KÖSZÖNETNYILVÁNÍTÁS 35

1. BEVEZETÉS

Napjainkban a mesterséges intelligencia nagy hatást gyakorol az életünkre. Rengeteg cég alkalmazza saját területeiken belül, mint például az egészségügyben, pénzügyekben, marketingben és az oktatásban. Mivel egyre pontosabbak, és jobbak döntéshozásban, így fontosságuk és pozitív hatásuk is egyre csak növekszik, megkönnyítik életünket.[1] Elég csak gondolni például az önvezető autókra, chatbotokra, digitális asszisztensekre és egyebekre. Érdemes kiemelni az OpenAI nevezetű cég által létrehozott ChatGPT chatbotot, ami szinte berobbant a közéletbe és hatalmas sikert aratott, ugyanis mindössze két hónappal elindítása után elérte a száz millió aktív felhasználót.[2] A rendkívül rövid idő alatt elért ”diadala” ösztönözte a technológia óriásait, köztük a Microsoftot, a Googlet és a Metat, hogy megalkossák a saját termékeiket, és felvegyék a versenyt a ChatGPT-vel.

A mesterséges intelligencia sokszor bebizonyította, hogy képes egyszerűbbé tenni feladatok széles tárházát, viszont ugyanilyen könnyedén fel lehet használni nem igazán nemes célokra is. A Deepfake-ek (Különböző médiák, amelyeket digitálisan manipuláltak, hogy meggyőzően helyettesítsék az egyik személy hasonlatosságát a másikéval), álhírek és káros propagandák valóságos futótűzként terjedhetnek az interneten és ezek gyártását a mesterséges intelligencia megléte nagymértékben megkönnyíti. Számos és egyre több olyan weboldal létezik, amik csak mesterséges intelligencia által generált hamis információkat magába foglaló cikkeket osztanak meg folyamatosan.[3] Nem csak weboldalak, hanem különböző szociális média platformokon lévő álprofilok és botok is léteznek, amik ezeket a tartalmakat osztják meg akár egyszerre többen is özönvízszerűen. Ilyen módon képesek felerősíteni egyes üzeneteket, az ellenvéleményeket és a tényeket pedig elnyomni. Ezek a szándékosan megtévesztő és manipulatív tartalmak rendkívül károsak és céljuk többek között a visszaélés, haszonszerzés és politikai befolyás szerzése például választások idejében.

A téves információkat különböző módszerekkel ugyan ki lehet szűrni és felismerni, de vannak esetek, amikor ez nem mindig sikerülhet, és mivel a mesterséges intelligenciák folyamatosan fejlődnek, így ezek szűrése egyre nehezebb kihívás. Szóval a mesterséges intelligencia segítségét élvező félretájékoztatás elleni harcban mi lehet az a megváltó eszköz, amit fel lehetne használni? Hát maga a mesterséges intelligencia. Ehhez olyan mesterséges intelligenciákat érdemes alkalmazni, amik be lettek tanítva egy hatalmas, nyers szövegeket tartalmazó adathalmazon, a szerzett tudásuk alapján pedig képesek mintákat felismerni, és

önállóan, emberi beavatkozás nélkül döntéseket hozni. Ezeket hívjuk „pretrained language model”-eknek, azaz előtanított nyelvi modelleknek.[4]

Ezeknek az előre tanított modelleknek a megléte tökéletes kiindulópontot ad nekünk, mert így elég csak a modell egyes paraméterein változtatni, és még tanítani a modellt új adatokkal, hogy egy specifikus feladat során minél jobb eredményt produkáljon. Ezt a módszert hívják finomhangolásnak. A mi esetünkben pedig a feladat, amire finomhangolni szeretnénk, az az úgynevezett szövegosztályozás, melynek lényege, hogy a modell az előre meghatározott kategóriákból képes legyen az adott szöveget a megfelelő kategóriába sorolni.

Ám a már hagyományosnak mondható finomhangolást egyre inkább felváltja a „prompt-tuning” technika, amely csupán annyit takar, hogy a bemeneti szöveghez hozzáadjuk a promptot, azaz a rávezető szöveget, és ezzel a modellt a kívánt kimenethez tereljük. Ez a módszer egy hatékonyabb és rugalmasabb alternatívát nyújt, viszont a megfelelő prompt szövegek kigondolása fejtörést okozhat sokaknak. Ezt a problémát hivatott megoldani az AutoPrompt nevű program, mely automatikusan képes promptokat generálni változatos feladatokhoz. Az AutoPrompt készítői szerint a programjuk segítségével bizonyítható, hogy ezek a modellek rendelkeznek azzal a képességgel, hogy finomhangolás nélkül meg tudják közelíteni a finomhangoláson átesett modellek teljesítményét.[5]

A leírtak alapján azt a tűztem ki célul, hogy ennek utána járok: Szakdolgozatomban a két technikát hasonlítom össze, és végezetül az eredmények alapján megállapítom melyik módszer hatékonyabb egy adott szövegosztályozási feladat elvégzése során.

2. FONTOSABB FELHASZNÁLT ESZKÖZÖK, KÖNYVTÁRAK BEMUTATÁSA

2.1. GOOGLE COLABORATORY

A módszerek összehasonlításához elengedhetetlen eszköznek bizonyult a Google Colaboratory vagy röviden csak „Colab”, a Google Research ingyenesen használható terméke, ami lehetővé teszi azt, hogy böngészőnkön keresztül írjunk, és futtassunk Python vagy R kódokat. Technikailag egy hosztolt Jupyter notebook szolgáltatás, ami nem igényel előzetes beállítást, és sok számítási erőforráshoz biztosít hozzáférést, köztük GPU-khoz is.[6] Érdemes viszont kitérni arra, hogy itt mi is az a notebook. A notebookok olyan dokumentumok, melyekben egyszerre találhatóak meg például számítógépes kódok, nyelvi leírások, vizualizációk (3D modellek, grafikonok, ábrák, stb.), és akár interaktív vezérlők is. A notebookokban írt kódokat képesek vagyunk futtatni is, kimenetüket pedig meg is tudjuk tekinteni.

A Colab különösen alkalmas gépi tanulási feladatokra és adatelemzésre, ezért a bevezetésben kitűzött célhoz egy tökéletes választás. A vizsgálatok során Pythonban írtam kódokat, illetve az ingyenesen használható T4 GPU lett felhasználva mint erőforrás azzal a céllal, hogy a kísérletek a lehető leggyorsabban végbemenjenek.

2.2. TRANSFORMERS KÖNYVTÁR

Egy másik, szintén nagyon fontos eszköz a Transformers könyvtár, ami nélkül nehezen sikerültek volna a kísérletek. A Transformers a HuggingFace nevezetű cég készítette, ami olyan számítógépes eszközöket fejleszt, amiket aztán gépi tanulást használó alkalmazások létrehozásához használhat fel bárki. A vállalat leginkább saját platformjáról nevezetes, amely lehetővé teszi a felhasználók számára, hogy megosszák saját gépi tanulási modelljeiket, és bemutassák munkájukat.[7]

A könyvtár rengeteg előtanított modellt tartalmaz, amikkel aztán feladatok sokaságát lehet elvégezni, mint például: kérdésmegválaszolás, szövegek összefoglalása, szöveggenerálás, objektumfelismerés, beszédfelismerés és a számunkra fontos szövegosztályozás. Emellett API-kat (Alkalmazásprogramozási felület) biztosít a modellek gyors letöltéséhez és használatához, finomhangolhatjuk saját adatainkon azokat, és munkánkat megoszthatjuk a közösséggel a cég saját platformján.

A Transformers mögött a három legnépszerűbb gépi tanulási keretrendszer áll: a Jax, a PyTorch és a TensorFlow. A keretrendszerek közt zökkenőmentes integráció áll fenn a Transformersen belül: modellünket egyszerűen betaníthatjuk az egyikkel, majd egy másikkal betölthetjük azt, hogy kiértékeléshez használjuk.[8]

2.3 PYTORCH, TORCH

A fentebb említett gépi tanulási keretrendszerek közül a PyTorchra esett a választásom, melynek alapját a Torch nevű könyvtár nyújtja. A Torch könyvtárt mesterséges neurális hálók készítéséhez alkalmazzák, de mik is azok a mesterséges neurális hálók?

A mesterséges neurális háló igazából egy biológiai ihletésű szimuláció, a valódi neurális hálózatok működését utánzó rendszer. A számítások alapegységeit hívjuk neuronoknak és maga a hálózat ezeknek a neuronoknak a rétegeiből áll fel. Az információ rétegről rétegre halad egy irányba a bemeneti rétegtől a kimeneti réteg felé.[9] A nyelvi modellek alapjait ilyen mesterséges neurális hálók nyújtják.

Visszatérve a PyTorch-ra, ez a keretrendszer kombinálja a Torchot egy Python alapú API-val. A Pytorch matematikai és programozási struktúrája leegyszerűsíti, és ésszerűsíti a gépi tanulási munkafolyamatokat anélkül, hogy limitálná a neurális hálók komplexitását és teljesítményét. Ezáltal lehetővé teszi azt használói számára, hogy kifinomult neurális hálózatokat építsenek ki és/vagy futtassanak miközben a kódra és matematikai szerkezetekre fordított idő minimalizálva van, jelentősen megkönnyítve a munkát, és közben hatékonyabbá is teszi azt. Népszerűségét elsősorban ennek köszönheti a PyTorch.[10]

2.4 DATASETS KÖNYVTÁR

Az adatok letöltését és azok előfeldolgozását jelentősen megkönnyítette a HuggingFace által készített Datasets könyvtár. A könyvtár segítségével több mint százezer adatkészlet érhető el a cég platformján és egy egyszerű, egysoros kóddal le lehet tölteni azokat majd felhasználni modellek tanításához és kiértékeléséhez, legyen az szövegekből, képekből vagy audio fájlokból álló adatkészlet.

Az ily módon elérhető publikus adatkészletekhez egyszerű és gyors előfeldolgozást segítő eszközök is társulnak a könyvtárban, amiket a saját számítógépünkön lévő adatokon is fel lehet használni, legyen szó bármilyen fájlkiterjesztésről: CSV, JSON, text, PNG, JPEG,

MP3, stb. Hasonlóan mint adat betöltés esetén adatainkat szimpla utasításokkal hatékonyan előkészíthetjük ellenőrzésre és modellek tanítására/értékelésére.[11]

2.5 EVALUATE KÖNYVTÁR

Finomhangolások után a modellek pontosságát az Evaluate könyvtár segítségével tudtam megtekinteni. Ez a könyvtár szintén a HuggingFace által készített eszköz, ami megkönnyíti a modellek értékelését és azok teljesítményeinek összehasonlítását.

A könyvtár segítségével több tucat népszerű metrikát lehet implementálni és még adatbázis-specifikus metrikákat is tartalmaz. Emellett még saját értékelési moduljainkat is létre lehet hozni és feltölteni a cég platformjára. Csakúgy, mint a Datasets könyvtár esetében, az Evaluate használata is rendkívül egyszerű: egysoros utasításokkal megadható milyen mérőszámok alapján szeretnénk értékelni a modellünket a kiértékelő kódunkban.[12]

2.6 AUTOPROMPT

A prompt-tuning módszer vizsgálásához az AutoPrompt nevű program nyújtott segítséget. Mint ahogy az a bevezetőben említve volt, a program célja, hogy automatikusan generáljon olyan promptokat, amelyekkel az adott modell a lehető legjobb eredményt éri el sok féle feladat során, így időt spórolunk meg, és még jobb eredményeket is érhetünk el a manuális prompt-tuninggal szemben.

Az AutoPrompt működésének alapja, hogy bármilyen természetesnyelv-feldolgozási problémát átalakít mondatbefejezési feladatokká, kiindulva egy általános sablonból: a sablon a bemeneti szöveggel kezdődik melyet úgynevezett ”trigger tokenek” követnek, amiket az AutoPrompt tanul meg, majd illeszt be, és végezetül az új szöveget egy ”predict token” zár, mely alapján lemérjük a nyelvi modell kimeneteit. A predict token helyét maga a modell tölti ki úgy, hogy előre megadott címkék közül választ egyet. Ezek a címkék kategóriákba vannak sorolva, és például szövegosztályozás esetén a bemeneti szöveget abba a kategóriába soroljuk, amelyik címkéit nagyobb valószínűséggel választja a modell. Ilyen módon egyszerűen, és paraméterek beállítása nélkül fel lehet mérni mire képes egy-egy nyelvi modell.[5]

A finomhangolással szemben ennek a programnak egy előnye, hogy ilyen módon egy darab modell is képes rengeteg feladatot teljesíteni sikeresen, és alkalomadtán jobban is teljesít, mint a hagyományosnak mondható finomhangolás.

3. FELHASZNÁLT ADATKÉSZLET BEMUTATÁSA

A bevezetésben nagy figyelmet kapott a félretájékoztatás és hogy milyen szerep jutott ebben a mesterséges intelligenciának, ezért a kísérletekhez felhasznált adatkészlet is az álhírterjesztés témájához kapcsolható.

Az elmúlt években különösen elvadultak az álhírek és pletykák a szociális médiában, viszont nem ok nélkül: a több ezer ember halálát okozó és országhatárok lezárását eredményező globális COVID-19 járvány megrázta az egész világot. Az Egészségügyi Világszervezet ”infodemicet” azaz információs járványt hirdetett: az igaz és téves információk sokasága nehezebbé tette az emberek számára azt, hogy megbízható forrásokat és útmutatásokat találjanak akkor, amikor szükségük van rá. Az ismeretlentől való félelem és bizonytalanság pedig nagy pszichológia befolyásoló erők az ember elméjében, és többször bebizonyosodott a történelem során, hogy az ijedt emberek olyan információkhoz ragaszkodtak a leginkább, melyek tudatában biztonságban érezhették magukat függetlenül attól, hogy igazak-e vagy sem. Épp ezért rendkívül nagy népszerűségnek örvendenek a különböző összeesküvés elméletek és álhírek.[13]

A felhasználásra került adatkészlet is a COVID-19 pandémiával foglalkozik. Ez a HuggingFace platformján publikusan elérhető[1], angol nyelvű és CSV formátumú adatkészlet 10700 darab szociális média bejegyzést és cikket tartalmaz, melyek vagy igaz, vagy hamis információkat tartalmaznak a járvánnyal kapcsolatban. Minden bejegyzésről lehet tudni, hogy valós vagy valótlan ugyanis meg vannak jelölve manuálisan a szerzők álltal. Az adatok három részből állnak: egy egyedi azonosítóból (”id”), a bejegyzésből (”tweet”) és a bejegyzés igazságtartalmából (”label”, ami lehet igaz vagy hamis).[14]

Az adatkészlet három részre van szedve: egy tanító, egy validáló és egy tesztelő készletre. Az elsőben 6420, a további kettőben pedig egyenként 2140 bejegyzés található. Ez a gépi tanulás területén egy bevett és hatékony technika a modell tanítása és kiértékelése során. A tanító készletet használjuk fel arra értelemszerűen, hogy a modellt betanítsuk az adatainkon felügyelt módon, azaz a bemeneti adat mellett a modell megkapja az elérni kívánt kimeneti eredményt is. A validációs készletet használjuk fel a modell hiperparamétereinek megválasztására, illetve tesztelésére a fejlesztés közben, így a tanítás közben fellépő hibákat időben lehet észlelni, és ezáltal azok elkerülhetővé válnak. Végezetül pedig a tesztelő készleten értékeljük ki a modell végső teljesítményét.

[1] https://huggingface.co/datasets/nanyy1025/covid_fake_news

4. KÍSÉRLETEK ELVÉGZÉSÉNEK MÓDJA

4.1. KÍSÉRLETEKHEZ FELHASZNÁLT MODELLEK

A technikák vizsgálatához három különböző korszerű előtanított nyelvi modell lett alkalmazva és összehasonlítva: a BERT[15], a RoBERTa[17] és a DeBERTa[19]. Mindhárom modell rendelkezik úgynevezett BASE és LARGE, azaz alap és nagy variánsokkal, tehát összesen hat modell lett felhasználva. A base modellek a large változatokhoz képest jelentősen kisebbek méretben, ugyanis a large variánsok több réteggel és paraméterrel rendelkeznek, emiatt több információt tudnak kinyerni a bemenetből, viszont éppen emiatt a tanulás erőforrásigényesebb, és más hátrányokat is magával hozhat. Mindhárom modell esetében a rétegmennyiség 12, míg ezeknek a nagy változatai 24 rétegből állnak. Paraméterszámok esetében a base BERT 110 millió, a large 340 millió paraméterrel rendelkezik, az alap RoBERTa 125 millió, a nagy variánsa 355 milliót foglal magába, végezetül az alap DeBERTa 100 millió és ennek nagy változata pedig 350 paramétert tartalmaz.

4.1.1 BERT

A Bidirectional Encoder Representations from Transformers (magyarul: Kétirányú kódoló-reprezentációk transzformátorokból), röviden BERT egy transzformátor architektúrán alapuló nyelvi modell melyet a Google mutatott be 2018 októberében. A transzformátor egy olyan gépi tanulási szerkezet, mely sorozatokat tartalmazó problémák megoldására alkalmas, ezek a sorozatok lehetnek szövegek vagy idősorok.

Minden transzformátor a következő fontos részekből áll össze: egy tokenizálóból, mely a bemeneti szöveget tokenekké konvertálja. Egy beágyazási rétegből, amely a tokeneket és a tokenek pozícióit átalakítja vektoros reprezentációra. A transzformátor rétegekből, amelyek a kapott vektoros reprezentációkon átalakításokat hajtanak végre ismételve így egyre több nyelvi információt (szöveges adatok esetén) kinyerve az adatokból, majd végezetül opcionálisan egy ”kiágyazó” rétegből, ami a végső vektor reprezentációkat a tokenek valószínűségi eloszlására alakítja vissza. A transzformátor rétegeknek kettő típusa lehet: kódoló és dekódoló. A BERT modell csak kódoló transzformátor rétegeket tartalmaz.

A BERT modell népszerűségét annak köszönheti, hogy jelentősen felülmúlta az akkor korszerűnek tekinthető modellek eredményét változatos benchmark feladatokon. Például a

GLUE (General Language Understanding Evaluation, azaz Általános nyelvértés kiértékelés) feladatgyűjteményen 80,5%-os teljesítményt ért el, ami 7.7%-al jobbnak mondható, mint az akkori nyilvántartott legjobb eredmény. Kiemelkedő teljesítménye betudható annak, hogy kétirányú módon tanul. Ez azt jelenti, hogy a tanulás során a bemeneti szöveget balról és jobbról párhuzamosan képes feldolgozni. Az így szerezett információk következtében a modell alaposan megérti a szöveg kontextusát.[15]

A BERT a fent említett okok miatt egy fontos kiindulóponttá vált a természetesnyelv feldolgozó kutatásokban és kísérletekben. Egy 2020-as felmérés szerint kicsivel több mint egy év alatt több mint 150, a modellt elemző és javító kutatási publikáció jelent meg.[16]

Fontos részlet még az, hogy a kísérletekhez a BERT cased változata lett felhasználva. Ez annyit tesz, hogy érzékeny a kis- és nagybetűkre, ezáltal különbséget tesz például a ”macska” és ”Macska” szavak közt.

4.1.2 RoBERTa

A Robustly Optimized BERT Pretraining Approach (magyarul: Robusztusan optimalizált BERT előképzési megközelítés), röviden RoBERTa egy olyan nyelvi modell, aminek alapja a Google BERT modellje melyet a Meta AI (akkoriban Facebook AI) mutatott be 2019 júliusában. Ez a modell a BERT-re úgy épít, hogy módosítja a kulcsfontosságú paramétereket, és nagyobb mini kötegekkel (köteg: egy tanulási iteráció során felhasznált tanuló adatok összesége) és sebességgel tanul.[17]

A RoBERTa készítőinek motivációjának számított a következő: azt állapították meg, hogy a BERT jelentősen alultanított volt, és ezért egy új tanítási módszert dolgoztak ki, amit később elneveztek RoBERTa-nak. Az elvégzett módosítások után a modell képes arra, hogy elérje az összes BERT-et követő modell teljesítményét, vagy azokat meghaladja.

A készítők legjobb modellje kiemelkedő eredményeket ért el különböző benchmark feladatgyűjteményeken: a GLUE-n 88.5-ös pontszámot ért el, megdöntve az akkori legmagasabb pontszámot, illetve más viszonyítási alapot nyújtó feladatokon is jól teljesített, például a SQuAD-on (Stanford Question Answering Dataset, magyarul Stanford kérdés válaszoló adatkészlet) és RACE-en (The ReAding Comprehension from Examinations, magyarul Olvasott szöveg értése vizsgálatokból) is.[18]

4.1.3 DeBERTa

A Decoding-enhanced BERT with Disentangled Attention (magyarul: Dekódolással továbbfejlesztett BERT szétválasztott figyelemmel) röviden DeBERTa egy Microsoft által fejlesztett nyelvi modell melynek célja, hogy a BERT és RoBERTa modellek teljesítményén javítson kétféle újszerű technika implementálásával. A modellt 2021 októberében mutatták be.

Az első technika, amit a készítők alkalmaztak, az az úgynevezett szétválasztott figyelem mechanizmus. A módszer lényege, hogy a bemeneti szövegben minden szó két vektorral van ábrázolva melyek kódolják annak tartalmát és pozícióját a szövegben, a szavak közti figyelem súlyok pedig a tartalmukra és a relatív pozíciójukra vonatkozó szétválasztott mátrixok segítségével vannak kiszámítva. A második technika pedig egy továbbfejlesztett maszk dekódoló, mely felváltja a kimeneti réteget azzal a céllal, hogy megjósolja a maszkolt tokeneket a modell előtanításához. (A maszk tokenek helyére a modell ”megjósolja” az oda legjobban illő szót.)

Ezekkel az alkalmazott módszerekkel a készítők elérték, hogy a modellek tanítása jelentősen hatékonyabb legyen a természetesnyelv megértésére és generálására irányuló feladatoknál. A DeBERTa fele annyi adatból képes volt arra, hogy a RoBERTa-t túlszárnyalja a természetesnyelv feldolgozó feladatok széles skáláján.[19]

4.2. FINOMHANGOLÁS MÓDSZER

A finomhangolási kísérletek a következőképpen lettek elvégezve: a kísérletek először a modellek base variánsain lettek teljesítve, majd azt követően a large változatokon. A felhasznált adatkészletben az egyedi azonosítók feleslegesnek számítanak, ezért ez az oszlop törlésre került.

Mivel a bejegyzések sokasága jelentős mennyiségben tartalmaz linkeket, emojikat és egyéb írásjeleket, ezért úgy gondoltam ki lehetne próbálni azt, hogy a bejegyzésekből bizonyos elemeket eltávolítok, és megvizsgálom hogyan teljesítenek a modellek ilyenkor, lesz-e bármilyen eltérés azzal szemben, mint amikor az adatok nem esnek át ilyen lépéseken. Igaz, hogy ezek a modellek toleránsabbak az ilyesmire, mert előtanításuk során találkoztak mindenféle szöveggel, viszont éppen ezért lesz ez érdekes számunkra, hátha akár rosszabb vagy jobb eredményeket produkálnak így a modellek. Ezek az elemek fokozatosan lettek eltávolítva vagy módosítva, tehát például először a nyers adatokon lett tanítva és tesztelve egy BERT

modell, majd azt követően az adatkészlet elemeiből törölve lettek például a linkek és aztán egy újonnan letöltött BERT modell pedig ezeken a szerkesztett adatokon lett tanítva majd kiértékelve, és így tovább. Az előfeldolgozási lépések a következő sorrendben lettek alkalmazva: kisbetűsítés, stopszavak törlése[2], URL-ek törlése, HTML speciális karakterek törlése (pl.: &amp, &gt), emojik törlése és végezetül az írásjelek eltávolítása.

Az adatkészlet átalakítása után következik az adatok tokenizálása az adott modell tokenizálójának segítségével. Tokenizálás során fontos, hogy a bejegyzések hossza ”csonkítva” legyen abban az esetben, ha az meghaladja a modell által megengedett maximális bementi hosszúságot, ami 512 volt minden kísérlet során. Ha ez a lépés nem történik meg, akkor a tanítás során a modell ezeket a hosszú adatokat nem fogja tudni kezelni és a tanítás félbeszakad.

Ezt követően a kiértékelő függvény létrehozására kerül sor. A mi esetünkben a legfontosabb metrika amit betöltünk, az a pontosság. Ebben a függvényben a modell predikcióit és a felhasznált szöveg kategóriákat adjuk tovább a pontosság kiszámításához.

Végezetül pedig elérkezünk a modell tanításához. Az alap tanítási argumentumok minden kísérlet esetében a következők: az értékelési és mentési stratégia paramétereknek az ”epoch” érték van megadva. Az epoch a gépi tanulásban az adatok egyetlen teljes áthaladását jelenti az adott algoritmuson belül. A kiértékelés és elmentés minden epoch végén fog végbe menni. Minden kísérlet során öt epoch fog lefutni, a futtatás végén pedig a validációs halmazon legjobban teljesítő modell lesz betöltve. Minden más paraméter az alapértelmezett értékével van ellátva. Ha a kísérlet túl sok erőforrást vesz igénybe, és emiatt hibák lépnek fel, akkor a következő lépések lesznek alkalmazva: a tanuló és értékelő kötegek mérete csökkentésre kerül, ha a hiba továbbra is fennáll, akkor a gradiensakkumulációs lépések száma lesz növelve. Gradiensakkumuláció során ugyanúgy kicsi gradiensekkel dolgozunk, de "összevárunk" belőlük néhányat, mielőtt a modellen frissítenénk. A gradiensek olyan értékek, amik segítségével az optimális eredményt szeretnénk elérni. Ha mindezek után se sikerül a problémát megoldani, akkor az úgynevezett gradiens ”ellenőrző pontozást” lehet alkalmazni. Ezzel több memóriát lehet megtakarítani, ami viszont azt eredményezi, hogy a számítási idő kis mértékben megnövekszik.

4.3. AUTOPROMPT MÓDSZER

[2] Olyan szavak törlése, amiket egy adott nyelvben gyakran használnak, ezáltal nem nyújtanak hasznos információkat. Ilyen szavak például a magyarban az ”a”, ”fel”, ”ki”, ”mi”, ”ott”, stb.

Az AutoPrompt program könnyedén letölthető a GitHub platformról. A program rendelkezik egy requirements.txt nevezetű szöveges fájllal, aminek tartalma olyan Python csomagok neveit és verziószámait tartalmazza, amik szükségesek az AutoPrompt használatához. Mivel a program legutoljára 2 éve lett frissítve, így ezeknek a csomagoknak a verziói elavultak lehetnek, így ezek a verziószámok átlettek írva a legfrissebb stabil kiadásokra.

A függőségek telepítése után következik az adatkészlet átformázása. Az AutoPrompt csak TSV illetve JSON kiterjesztésű fájlokat támogat, ezért az általam használt adatkészletet TSV formátumú fájllá kell alakítani, viszont itt nem áll meg a szerkesztés. A program nem nyújt segítséget abban, hogy a tokenizálás során a túl hosszú bemeneteket rövidítsük, ezért a program futtatása előtt még ezeket a bejegyzéseket vagy töröljük, vagy manuálisan lerövidítjük. Az adatkészlet mindhárom részében megvizsgálom az adatok hosszát, majd ezeket tokenizálom az adott modell tokenizálójával. Amelyik bejegyzés tokenjeinek száma meghaladja a modell maximálisan megengedett bemeneti méretét, azokat töröljük abban az esetben, ha ez elenyészően kevés adatot érint. Mivel a teljes adatkészletben mindössze öt darab bejegyzés hosszabb, mint a maximális méret, ezért ezek törlésre kerültek.

Ezek után elérkezünk a program futtatásához. A promptok generálásához a create_trigger.py fájl kerül futtatásra, a minden kísérlethez felhasznált paraméterek a következők: A sablonunk, amit felhasználunk így néz ki: {sentence} [T] [T] [T] [P]. Itt a {sentence} mező a bemeneti szöveg, a [T] mezők helyére pedig a program illeszt majd be szavakat, amikkel a modellt sikeresen a megfelelő kimenet felé lehet terelni, ezek lesz a trigger tokenek. Végezetül a [P] helyére a modell illeszt be egy oda szerinte megfelelő szót az általunk megadott címkékbe és az azokba sorolt szavak közül. A felhasznált címkék természetesen a ”real” és a ”fake”, azaz igaz vagy hamis. Ezekbe a kategóriákba öt-öt szó került, amiket én gondoltam ki. A ”real” kategória a "tests", "results", "testing", "reports", "cases"; a ”fake” pedig a "vaccine", "masks", "brainwash", "man-made" és a "force" szavakat tartalmazza. A trigger token jelöltek száma százra van lekorlátozva. Az AutoPrompt a tanító adatkészletet használja fel a megfelelő promptok előállításához és kiszámítja melyik trigger token jelöltek okozzák a legnagyobb változást a modell pontosságában.

Minden modell és azok változatai esetében kilenc különböző kísérlet kerül végrehajtásra. A kilenc kísérlet háromféle kategóriába sorolható: Az első kategóriában minden vizsgálatban öt iteráció megy végbe, a másodikban tíz iteráció, a harmadikban megint csak öt iteráció fut le, viszont az alapértelmezett tíz akkumulációs lépés helyett húsz lépés történik. A

kategóriákon belül a három kísérlet között annyi a különbség, hogy mindegyik más-más tanuló és értékelő kötegméreteket használ.

5. EREDMÉNYEK, KÖVETKEZTETÉSEK

5.1. FINOMHANGOLÁS EREDMÉNYEK

5.1.1. BERT-BASE-CASED

Az adatok előfeldolgozása nélkül a modell a második epochban teljesített a legjobban, 0,9776-es pontosságot és 0,0941-es veszteséget elérve a validációs halmazon. A veszteség azt jelzi, hogy a modell által hozott döntés és a tényleges célértékek közötti eltérés mekkora egy adott adatkészleten, esetünkben a validációs halmazon. A veszteség értékét a kategorikus kereszt-entrópia veszteségfüggvény segítségével kapjuk meg. Minél kisebb ez az érték, annál kisebb az eltérés, tehát a modell pontosabb, ami nekünk jó.

Az adatok kisbetűsítése után a modell teljesítménye picivel rosszabb lett, a negyedik epochban elért 0,9748-es pontosság még nem is jelentős eltérés, viszont a 0,1628-es veszteség majdnem a duplája az előzőnek. A stopszavak törlését hozzáadva a kisbetűsítéshez tovább rontott kismértékben a teljesítmény: 0,9664-es pontosságot és 0,1694-es veszteséget produkált a BERT a tanulás harmadik epochjában. Ezt követően az URL-ek kerültek törlésre és a modell teljesítménye tovább romlott, a pontosság 0,9565-re csökkent, a veszteség pedig 0,1707-re nőtt a második epochban. A HTML speciális karakterek törlése után viszont a pontosság az előzőhöz képest javult, 0,9621-et elérve, de a veszteség csak tovább nőtt, most már 0,2291-et elérve a negyedik epochban. Az emojik törlését hozzá adva a feldolgozási lépésekhez egy újabb érdekességet nyújtott. A pontosság terén a második epochban 0,957-et ért el a modell, de a veszteség 0,1913-ra csökkent. Végezetül az írásjelek törlése után az első epochban a pontosság kirívóan 0,9318-re csökkent, a veszteség pedig 0,2058-ed lett.

Az eredményekből jól látható, hogy az adatok feldolgozása nélkül éri el a modell a legjobb teljesítményt, pontosság és veszteség terén is. A legrosszabb pontosság az összes feldolgozási lépés alkalmazása után lett elérve, a legrosszabb veszteség pedig a HTML karakterek törlése után történt.

5.1.2 ROBERTA-BASE

A RoBERTa alap variánsa előfeldolgozás nélkül 0,9804-es pontosságot és 0,0956-es veszteséget ért el a harmadik epochban, ami a legjobb BERT base modellhez képest jobb

pontosságot produkál, viszont alulmarad annak veszteségéhez képest. Kisbetűsítés után a pontosság és veszteség romlott, csakúgy, mint a base BERT modell esetén. A pontosság a harmadik epochban 0,9776-re csökkent, a veszteség pedig 0,1273-re nőtt. A helyzeten csak tovább rontott a stopszavak törlése, viszont csak kis mértékben, a pontosság és veszteség szintén a harmadik epochban mostmtár 0,9706 és 0,137. A BERT alap modellhez képest jelentősen eltérő eredményeket produkált az URL-ek törlése a bemeneti adatokból, ugyanis a pontosság 0,9369-re esett le, a veszteség pedig 0,2102-re növekedett megint csak a harmadik epochban. A pontosság területén azonban javított a HTML karakterek törlése, ami így feljebb mászott a 0,9495 értékre, de a veszteség tovább nőtt 0,2229-re mostmár a negyedik epochban. Miután az emojik is törölve lettek az adatkészletből a modell 0,9528-es pontosságra és 0,16-os veszteségre volt képest a második epochban, amik jobbak az előző két kísérlethez képest, de az URL-ek törlése előtti kísérletek eredményeit nem éri el. Az utolsó szöget a koporsóba az írásjelek törlése biztosította, ugyanis újabb drasztikus romlás következett be: Az utolsó, ötödik epochban a pontosság 0,8949-re csökkent, amivel először estünk be 0,9 alá, a veszteség 0,3275-re nőtt, itt pedig először értünk el 0,3 feletti értéket.

Csakúgy, mint a BERT-nél, az előfeldolgozási lépések nem értek el semmilyen javulást, viszont a teljesítmény hanyatlása itt jóval nagyobb mértékűre sikeredett. A legjobb eredményt természetesen feldolgozás nélkül, a legrosszabbat pedig minden lépés elvégezte után értük el.

5.1.3 DEBERTA-BASE

A következő kísérletek során alkalmazni kellett pár paraméter módosítást, ugyanis a vizsgálatok közben memóriával kapcsolatos hibák léptek fel, ezért a tanuló és értékelő kötegek mérete az alapértelmezett nyolcról négyre lett csökkentve.

Előfeldolgozás nélkül a modell 0,9748-es pontosságot és 0,1573-es veszteséget ér el az ötödik epochban. A BERT és RoBERTa modellekhez képest ez a veszteségi érték jóval nagyobb. Kisbetűsítés után ugyanaz a tendencia figyelhető meg, mint az előző két modell esetén, a pontosság 0,9678 és a veszteség 0,1673 a negyedik epochban, stopszavak törlése után pedig 0,9612 pontosságot és 0,2102 veszteséget érünk el az ötödik epochban. URL-ek törlése után egy kisebb ugrás történik a rossz irányok felé: Az ötödik epochban a 0,9383 lett a pontosság, a veszteség pedig 0,2917. HTML speciális karakterek törlése után a második epochban a pontosság 0,9 alá esik, a veszteség pedig meghaladja a 0,3-et, a pontos adatok 0,8958 és 0,3753. Az emojik törlése itt is javított újra a teljesítményen, csakúgy, mint a

RoBERTa esetén: a pontosság 0,9463-re nőtt, a veszteség pedig 0,2527-re csökkent vissza az ötödik epochban. Végezetül pedig az írásjelek törlése megint csak tovább rontott a teljesítményen, hasonlóan a két előző modell vizsgálatai során látottakhoz. Az első epochban a pontosság 0,8374 lett, a veszteség pedig 0,4681-re nőtt, majdnem elérve a 0,5-et.

Megint csak bebizonyosodott, hogy az előfeldolgozás nem javít a modellek teljesítményén, és megint akkor teljesítenek legrosszabban, ha minden feldolgozási lépés el lett végezve. Éppen ezért a következő kísérletek során a nagy modell variánsokon ezek az előfeldolgozási lépések nem lesznek megvizsgálva, kihagyásra kerülnek.

A DeBERTa modell egy extra kísérleten is átesett előfeldolgozás nélkül. Ebben az esetben egy újabb paraméter lett átállítva: a gradiens akkumulációs lépések száma az alapértelmezett egyről kettőre lett állítva. Ekkor a modell pontossága a második epochban 0,9752, vesztesége pedig 0,1432 lett. Ezzel az egy paraméter beállítással javítani tudtunk a DeBERTa előző legjobb eredményein.

5.1.4 BERT-LARGE-CASED

A BERT nagy változata esetén is a kötegméretek négyre lettek állítva memória hibák miatt. A finomhangolás futtatása során érdekes dolgok történtek. A tanulás során minden epochban a tanulási veszteség mindig 0,7 körül, a validációs veszteség pedig 0,69 körül maradt, a modell nem volt képes tanulni az adatokon. A pontosság egy epoch kivételével mindig 0,5234, a legjobb veszteség pedig 0,6921 volt az ötödik epochban.

Ezt a jelenséget viszont kiküszöbölte a gradiens akkumuláció kettőre állítása. A validációs veszteségek az epochokban mostmár 0,1 körül voltak, a legkisebb 0,1237 volt a harmadik epochban. Pontosság terén is jelentős javulás történt, az előzőleg említett veszteség mellet a pontosság 0,9743 lett. Ez a teljesítmény viszont még nem haladja meg az előző base modellek legjobb eredményeit.

Ebből a két kísérletből jól látható, hogy a finomhangolás eredményei nagyon tudnak szórni és erre a nagy modellek kifejezetten hajlamosak tudnak lenni. Ezt viszont javíthatja egy stabilizációs eljárás, mely során jó pár paraméter átállításra kerül annak érdekében, hogy a large modell változatok stabilabban viselkedjenek finomhangolás során.[20]

A harmadik és egyben utolsó large BERT finomhangolási kísérletben a következő paraméterek lettek megváltoztatva: az AdamW optimalizáló tanulási sebessége 3e-5-re (azaz $3x10^{-5}$) lett állítva. Az AdamW egy olyan optimalizációs módszer, amit a modellekben

használnak fel finomhangolás során, hogy csökkentsék a tanulási veszteséget. A bemelegítési arány 0,1-re lett beállítva. Itt a bemelegítés azt jelenti, hogy nem egyből a megadott tanulási rátával kezdjük a finomhangolást, hanem fokozatosan növeljük azt. Esetünkben a bemelegítés 0,1-es aránnyal történik és fokozatosan megy felfele az a tanulás során, amíg el nem éri a megadott tanulási sebességet. Az AdamW optimalizáló epszilon paramétere 1e-6-ra lett állítva, a béta egyes és kettes paraméter pedig változatlanul 0,9 illetve 0,999 maradt. Végezetül a súlycsökkentés 0,01-ra lett beállítva, a maximális gradiens norma pedig megmaradt 1-nek.

Ezek a lépések javítottak a modell teljesítményén: a legjobb eredmény 0,9827-es pontosságot hozott és 0,102-es veszteséget a harmadik epoch során. Ezek az eredmények két érdekességet hoztak magukkal: Az eddigi base modelleknél jobb pontosságot sikerült elérni, viszont a veszteség csak a DeBERTa modellénél lett jobb.

5.1.5 ROBERTA-LARGE

A következő tesztelésre kerülő modell a RoBERTa nagy variánsa. Az előző modellekhez hasonlóan itt is négyre lettek állítva a kötegméretek a memória hibákba való ütközések miatt. Az első kísérlet során minden epochban a tanulási veszteség csökkent ugyan, de csak 0,6693-et sikerült csak elérni, a validációs veszteség pedig egyre csak növekedett, és végül elérte az 1,57-ot. A kapott modell tehát nagyon rossz lett. A legjobb eredmény a második epochban történt, a pontosság 0,4766 lett, a veszteség pedig 0,6947. Ez borzasztó teljesítmény az előző kísérletekhez képest.

Ezen a teljesítményen sajnos a gradiens akkumuláció kettőre állítása se segített nagymértékben. A pontosság és veszteség minimálisan javult: 0,5234-et és 0,6921-et sikerült csak elérni a harmadik epochban. Itt is a BERT-hez hasonlóan képtelenek leszünk stabilizációs lépéseket elvégezni, hogy valamilyen értékelhető teljesítményt érjünk el. Ami érdekes viszont, hogy ennek a kísérletnek a pontossága összeadva az előző kísérlet pontosságával pontosan 1,0. A validációs halmazban a címkék eloszlása megegyezik a kísérletek pontosságaival: a 0,4766 a „hamis” címkék aránya, míg a 0,5234 az „igaz” címkék aránya. Ez azt jelenti, hogy a modell az ezt megelőző vizsgálatban beállt arra, hogy mindenre a „hamis” címkét adta válaszul, míg ebben a kísérletben a „igaz” címkével válaszolt meg minden bemenetet.

A RoBERTa nagy változata esetén a következő paraméterek lettek átállítva: A tanulási sebesség 3e-5-re, a bemelegítési arány 0,1-re, Az AdamW epszilonja 1e-6-ra, a béta egyes 0,9 maradt viszont a béta kettő 0,98-ra lett átállítva. Végül a súlycsökkentés pedig 0,1-re lett

beállítva. Ezek a lépések eredményesnek bizonyultak, ugyanis az ötödik epochban sikerült elérni a 0,9 feletti pontosságot, a pontos legjobb érték 0,9668. Ami a veszteséget illeti, a többi modellhez képest alulmarad, ennek értéke csupán 0,1757 lett, viszont ez fényévekkel jobb, mint az előzőleg elért 0,6921.

Összeségében ezek a stabilizáló lépések eddig jól működnek ezeken a modelleken, most már csak a DeBERTa nagy variánsának kiértékelése maradt hátra.

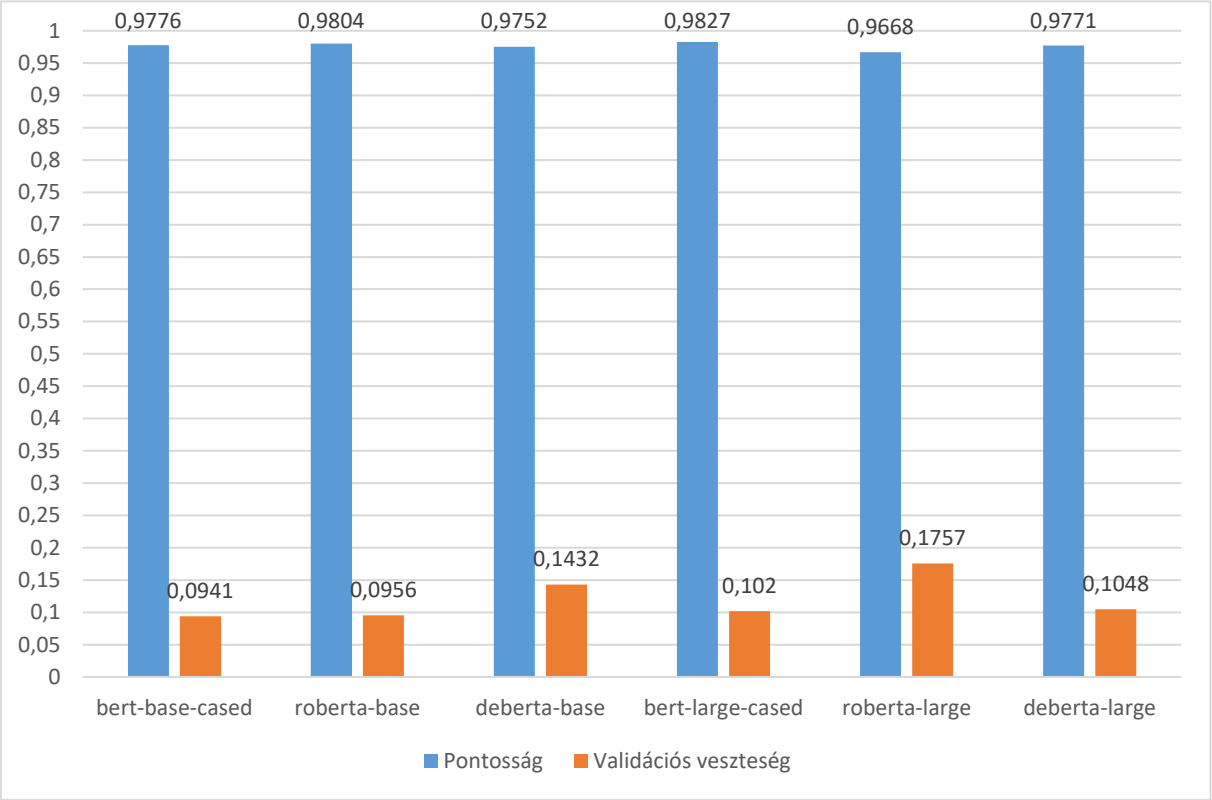
5.1.6 DEBERTA-LARGE

A kötegek nagyságát itt is állítani kényszerültem, viszont a jól bevált négyes nem bizonyult elégnek, a hibák ugyanúgy felütötték fejüket, szóval a kötegek mérete kettőre lett csökkentve. Ennek köszönhetően megtörtént az eddigi leghosszabb kísérlet, mely egy óra és harmincöt percig tartott. Türelmes várakozás után érdekes eredményeket véltem felfedezni. A BERT-hez és RoBERTa-hoz képest jobban teljesített a DeBERTa akkor, amikor csak a kötegméretek lettek átállítva. 0,7972-es pontosságot és 0,5093 validációs veszteséget sikerült elérni az ötödik epochban, ami felülmúlja a BERT 0,5234-es pontosságát, 0,6921-es veszteségét, a RoBERTa 0,4766-es pontosságát és a 0,6947-es veszteségét. Viszont ahogy azt már láthattuk ennek a két modellnek a teljesítményét sikerült javítani, ezért nézzük most meg azt, hogy sikerül-e a DeBERTa-nak lépést tartania.

Eddig a gradiensakkumuláció mindig valamilyen mértékben javítani tudott a modelleken, ezért érdemes ezt most is megvizsgálni. A gradiens ”ellenőrző pontozás” segítségével a kötegméreteket visszaállíthatjuk négyre, a gradiensakkumulációs lépések száma pedig ismételten kettőre lett állítva. A modell így az első epoch során 0,8547-es pontosságot és 0,4159-es veszteséget ért el, ami ugyan jobbnak bizonyul a RoBERTa-énál, de annak végső teljesítményét még nem haladja meg.

Az eddig nagyon jól bevált stabilizációs lépések felhasználása következik. Ugyanazok a paraméterek kerültek átállításra, mint amik a BERT optimalizálása során ugyanazokkal az értékekkel. Ez az eljárás újra bebizonyította, hogy nagyon értékes, ugyanis 0,9771-es pontosságot értünk el 0,1048-es veszteséggel a második epochban. Ezzel az eredménnyel a DeBERTa nagy variánsa megközelítette a nagy BERT-et és felülmúlta a RoBERTa nagy változatának teljesítményét.

5.1.7 FINOMHANGOLÁSI EREDMÉNYEK ÖSSZEHASONLÍTÁSA



5.1: A modellek legjobb teljesítményei finomhangolással

A kapott eredmények sokkal jobbnak bizonyulnak egy olyan alapmodell pontosságához képest, ami a többségben lévő címkét választja minden bemenet esetén, ami az „igaz” címke, az érték pedig 0,5234. A base modellek között a legjobb pontosságot a RoBERTa érte el, viszont a legkisebb veszteséget a BERT produkálta. Itt a DeBERTa nem tudott kiemelkedni: sikerült megközelítenie a BERT pontosságát ugyan, de a vesztesége alulmarad jelentősen a többihez képest. A large modellek esetében a BERT-nek sikerült elérnie a legjobb pontosságot és a legkisebb veszteséget. Ezt követte a DeBERTa, majd a RoBERTa, ami a legrosszabb teljesítményt produkálta, rosszabbat bármelyik alap modellhez képest.
Összehasonlítva mindkét variánst, az alap BERT és RoBERTa veszteségét egyik nagy modell se érte el, viszont a legjobb alap modell pontosságot meghaladja a BERT nagy változata.

5.2 AUTOPROMPT EREDMÉNYEK

5.2.1. BERT-BASE-CASED

Az AutoPrompt program a kísérletet megelőzően ráengedi a validációs adatkészletre a modellünket, és megvizsgálja annak pontosságát. A BERT 0,5213-es pontosságot ért el finomhangolás és prompt-tuning nélkül. Az első három kísérlet során, mint ahogy az a negyedik fejezetben ismertetve lett, öt iterációs a futtatás tíz akkumulációs lépéssel. Először a kötegméretek nyolcra lettek beállítva. Ekkor a modell 0,7288-es pontosságot ér el a kiértékelő adatkészletünkön. Tizenhatos kötegméretek mentén 0,7017 lett a pontosság, ez rosszabb az előző eredményünkhöz képest. Huszonnégyes kötegméreteknél a program csak 0,6854 pontosságot volt képes elérni.

A következő három kísérletnél az eddigi öt helyett tíz iteráció megy végbe. Ugyanúgy, mint az öt iterációs futtatás esetén 0,7288-es pontosságot produkált a modellünk nyolcra beállított kötegméreteknél. A következő kísérlet esetén is ugyanaz lett az eredmény, 0,7017 lett ismét a pontosság tizenhatos kötegméreteknél. A harmadik vizsgálat se tartogatott meglepetéseket, a pontosság 0,6854 lett huszonnégyes kötegeknél is.

A legutolsó három kísérletben visszatérünk az öt iterációs futtatáshoz, de az akkumulációs lépések húszra lettek állítva. A kapott eredmény nyolcas kötegeknél 0,7017 lett, ugyanaz, mint amit az előző tizenhatos kötegméretes kísérleteknél értünk el. Az ezt követő kísérlet 0,6853-es pontosságot hozott tizenhatos kötegeknél, ami pedig megegyezik az eddigi huszonnégy kötegméretes vizsgálatok eredményeivel. Az utolsó kísérlet sajnos nem tudott sikeresen lefutni, az első iterációig se jutott el a program, a hibát nem sikerült kiküszöbölni.

A legjobb eredményt először öt iterációval, nyolcas köteg méretekkel és tíz akkumulációs lépésekkel értük el.

5.2.2 ROBERTA-BASE

A RoBERTa 0,6101-es pontosságot ér el alap állapotában, nézzük meg, hogy ezen az AutoPrompt tud-e javítani.

Az öt iterációs és tíz akkumulációs lépéses kísérletek a következő képpen alakultak: 0,7284 lett a pontosság nyolcas méretű kötegekkel, tizenhatos kötegeknél 0,6358-es pontosságot sikerült csak elérni, ami egy drasztikus változás az előző vizsgálattal összehasonlítva. Végezetül a huszonnégyes kötegméretek esetében a pontosság 0,7349 lett, ami a legelső eredményünket meghaladja. Nagy kiugrások történtek tehát, hátha a következő kísérletek is hasonló izgalmakat tartogatnak.

Tíz iterációs futtatásoknál a nyolcas kötegeknél ugyanazt a teljesítményt nyújtotta a modell 0,7284-es pontossággal. Tizenhatra állított kötegeknél a modell megint csak alulmaradt, most viszont kisebb mértékben. Az elért pontosság 0,6774 lett. Lezárásként újra 0,7349-es pontosságot ért el a modell huszonnégyes kötegekkel.

Az utolsó kísérletek közül az első 0,6358-et produkált, ami megegyezik az tíz akkumulációs és tizenhatos kötegméretes kísérlet eredményével. Tizenhatos kötegekkel 0,7265 lett az eredmény, huszonnégy esetén pedig 0,6652-es pontosságot sikerült elérni.

A RoBERTa a legjobban huszonnégyes kötegekkel, öt iterációval és tíz akkumulációs lépéssel teljesített.

5.2.3 DEBERTA-BASE

Az utolsó base modell, amit vizsgálunk megint csak a DeBERTa, ami önmagában 0,5231-es teljesítményre képes. Az első kísérletek így alakultak: 0,5386-es pontosságot értünk el nyolcas kötegmérettel, ami nagyon kicsi javulás 0,5231-hez képest. Tizenhatos mérettel 0,5241 lett az eredmény, ami az előzőnél sokkal kisebb javulás. A következő alkalmazott kötegméret a tizennégy, ugyanis minden tizenhat feletti méret esetén a program futása megszakad memória hibák miatt. Ebben az esetben 0,5241-et sikerült elérni, ami megegyezik az ezt megelőző kísérlet pontosságával.

A tíz iterációs kisérleteknél a maximum kötegméret tizenkettő ismét memória hibák elkerülése miatt. Nyolcas méretű kötegekkel 0,5647-es pontosságot produkált a modellünk, ez az eddigi legjobb eredmény. A következő kísérlet ezt meg is döntötte tizenkettes mérettel, a pontosság 0,6054 lett. Tízes kötegekkel a DeBERTa teljesítménye 0,605 lett, csupán csak négy tízezreddel maradt alul az előző kísérlet pontosságához képest.

Húsz akkumulációs lépéssel és öt iterációval a következő eredményeket kaptuk: az első kísérlet során 0,5241-et kaptunk nyolcra állított kötegekkel, a másodiknál újra 0,5240-es pontosságot értünk el tizenkettes kötegméretekkel, végezetül pedig tízes kötegekkel 0,5568-et sikerült összehoznia a programnak.

Összeségében ez a kilenc kísérlet nem bizonyult igazán sikeresnek, a legjobb eredményt tizenkettőre beállított kötegméretekkel, tíz iterációval és tíz akkumulációs lépéssel kaptuk meg.

5.2.4 BERT-LARGE-CASED

Elérkeztünk a modelljeink nagy változataihoz, amikkel a finomhangolásban kissé meggyűlt a bajom azok szórása miatt a finomhangolási kísérletek során. Most azonban emiatt nem kell aggódni remélhetőleg az AutoPrompt használata során.

Minden kísérlet során nyolc, tizenhat és huszonnégyes kötegméretek lettek alkalmazva, kivéve az utolsó három vizsgálat esetében, ahol nyolc, tizennégy és tizenhat lett használva memória hibák miatt. A BERT large variánsa 0,467-es pontosságot ért el az előtanított, alap formájában, ami rosszabb a base változat teljesítményéhez képest.

Az öt iterációs vizsgálatok eredményei a következők: Nyolcas kötegekkel 0,7821-es lett a pontosság, tizenhatosnál 0,7237-et ért el a program, huszonnégyes mérettel pedig a pontosság 0,719 lett. Jól megfigyelhető, hogy egyre nagyobb kötegméretekkel a pontosságunk romlott folyamatosan.

A tíz iterációs kísérletek nem igazán tartogattak újdonságokat. Nyolcra állított kötegméretekkel ismét 0,7821-es pontosságot ért el a BERT, tizenhat esetében 0,7237 lett a pontosság és végezetül a huszonnégyes kötegekkel megint 0,719-et kaptam. Minden kísérletünk megegyezik az előzőek eredményeivel.

Nézzük meg most azt, hogy húszra állított akkumulációs lépésekkel mit tartogat számunkra a nagy BERT. Az első vizsgálat során 0,7237-es pontosságot sikerült elérnie a programnak. Ezt a számot már láttuk korábban a tizenhatos kötegméretes kísérleteknél. Tizennégyes kötegekkel 0,7223-et hozott ki a modellből az AutoPrompt és ezt lezárva tizenhatra állított kötegméretekkel 0,7307-es pontosságot kaptunk.

A legjobb teljesítményt legelőször öt iterációs, nyolcas kötegméretes és az alapértelmezett tízre állított akkumulációs lépéses futtatással kaptuk meg.

5.2.5 ROBERT-LARGE

A RoBERTa nagy modelljének vizsgálata során nyolcra, tizenhatra és huszonkettőre állított kötegeket használtam, kivéve a húsz akkumulációs lépéses vizsgálatoknál, ahol a huszonkettő helyett húszat alkalmaztam. Az előtanított modell 0,6031-es pontosságot ért el a kiértékelő adatkészletünkön, ami kb. egy századdal marad alul az alap RoBERTa modellhez képest.

Az öt iterációs kísérletek alatt elért teljesítmények a következők: Először 0,7242-et hozott össze a program, ezt követően 0,7087 lett a pontosság, az utolsó vizsgálat pedig 0,6872-et produkált a RoBERTa az AutoPrompt segítségével. Itt is jól látható a teljesítmény romlása egyre nagyobb és nagyobb kötegek mentén.

Tíz iterációs vizsgálatoknál csak úgy, mint a BERT esetében az eredmények megegyeztek az előző kísérletek által előállított számokkal. A pontosságok tehát ismét 0,7242, 0,7087 és 0,6872 lett.

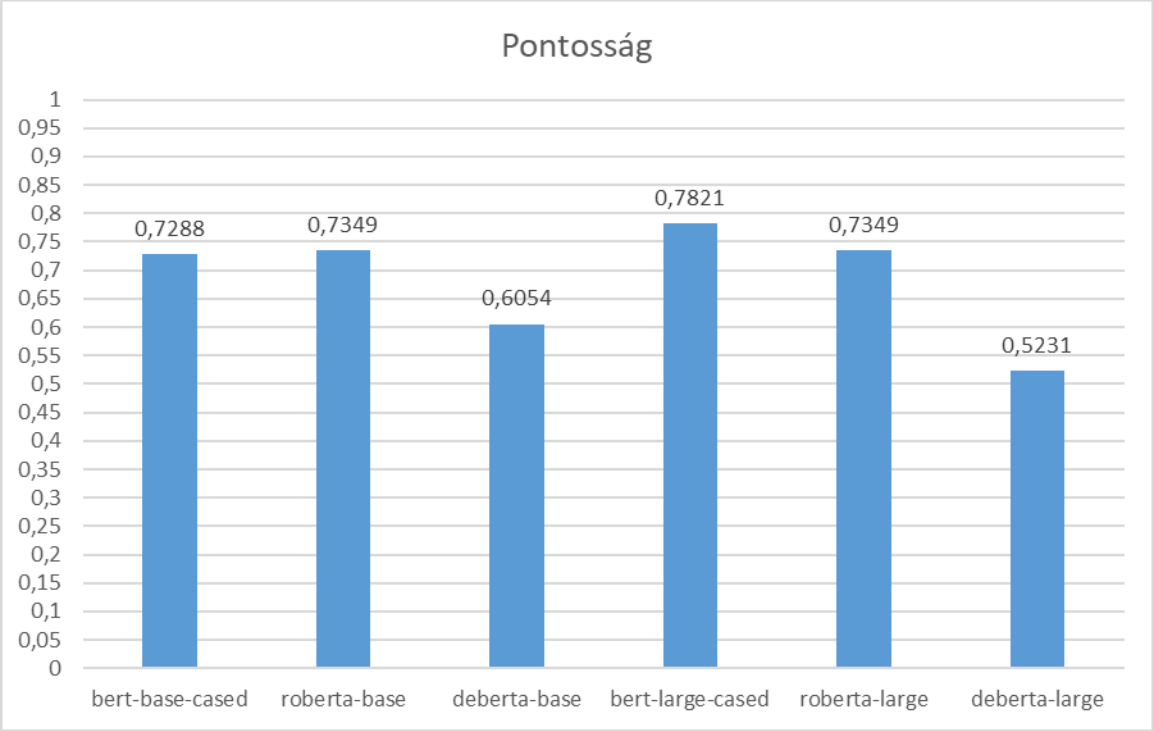
A teszteket lezáró hármas a következő értékeket eredményezte: Először 0,7087-et értünk el, ami egyenlő az előző tizenhat kötegméretes vizsgálatok eredményeivel. A második teszt 0,7349-es pontosságot produkált és a végső kísérlet során pedig 0,69-et hozott össze a program.

A nagy RoBERTa-t tesztelő kísérletek közül a legjobb teljesítményt a tizenhatra állított köteges, öt iterációs és húsz akkumulációs lépéses vizsgálat hozta. Az AutoPromptos legjobb kísérleteink során ez az első olyan, aminél az akkumulációs lépések száma húszra lett állítva.

5.2.6 DEBERTA-LARGE

A felhasznált kötegméretek a kísérletek során a nyolc, tíz és tizenkettő az ekkor már jól ismert memóriával kapcsolatos hibák megelőzése miatt. Promptok nélkül a DeBERTa nagy változata 0,5231-es pontosságot ért el. Ezt a számot a kilenc kísérlet során nem sikerült egyáltalán meghaladni, az AutoPrompt nem talált olyan promptokat amikkel javítani tudta volna a modell teljesítményét, hiába futottak a vizsgálatok hosszú időn keresztül.

5.2.7 AUTOPROMPT EREDMÉNYEK ÖSSZEHASONLÍTÁSA, LEGJOBB MODELL TOVÁBBI TESZTELÉSE



5.2: A modellek legjobb teljesítményei az AutoPrompt használatával

A csak az „igaz” címkét választó alapmodellünknél mindegyik modell jobban teljesít, kivéve a DeBERTa nagy változata, ami csupán három tízezreddel marad alul az alapmodellünkhöz képest. A legjobb eredményt elérő alap modellünk a RoBERTa lett a maga 0,7349-es pontosságával, a nagy modellek közül pedig a BERT lett 0,7821-es teljesítménnyel. Megfigyelhető, hogy a base és large RoBERTa modellek teljesítményei megegyeznek. A DeBERTa modellek láthatóan alulmaradnak a többi nyelvi modellhez képest. A látottak alapján kijelenthető, hogy a nagy BERT a legjobb modellünk, amivel további kísérletek lettek elvégezve.

A program további két alkalommal, ugyanazokkal a paraméterekkel lett futtatva, az egyetlen dolog, ami meg lett változtatva az a random seed, amely alapján az osztályozó réteg súlyainak kezdőbeállítása megtörténik. A random, azaz véletlenszerű seed egy olyan szám, ami segítségével a kísérletek során elért eredményeink reprodukálhatóak lesznek. Az eddigi vizsgálatok során az alapértelmezett random seedet használtam, ami a nulla.

A két különböző random seed-el ellátott kísérlet 0,7209-et és 0,7101-et ért el, amik egyértelműen kisebbek a 0,7821-nél. A legjobb modellünk átlagos teljesítménye így 0,7377, ami még mindig jobb a többi modell legjobb eredményéhez képest.

A következő négy kísérletben az általam megadott címkékbe sorolt szavak lettek megváltoztatva. Minden vizsgálat során különböző öt-öt szó lett alkalmazva. Az első szó

kombinációval 0,6891-et produkált a modell, a másodikkal 0,6601-et, a harmadikkal 0,7022-et, végezetül a negyedikkel pedig 0,7274-et ért el. A kapott eredmények alapján az először megadott, minden modellnél alkalmazott szavak bizonyultak a legjobbnak, az előzőleg tesztelt négy új szó kombináció nem hozott javulást.

5.3 MÓDSZEREK ÖSSZEHASONLÍTÁSA

Az elvégzet kísérletek azt mutatták meg, hogy a hagyományos finomhangolás jelentősen felülmúlja az AutoPrompt által produkált teljesítményeket. A finomhangolás során minden modell legjobb teljesítménye meghaladja a 0,9-et, míg az AutoPrompt esetében az értékek a 0,8-et se érik el.

6. ÖSSZEFOGLALÓ

A szakdolgozatom célja az volt, hogy az AutoPrompt módszert összehasonlítsam a hagyományos finomhangolással egy álhírfelismerési feladat során, és ezalatt elérjem azt, hogy az AutoPrompt megközelítse a nyelvi modellek osztályozási teljesítményét, amit finomhangolással érünk el. Ez a kitűzött cél többé-kevésbé el lett érve. A BERT és a RoBERTa modellek esetén a pontosságok meghaladják a 0,7-et, ami egy reális és elfogadható kimenet a modellektől. Persze alulmaradnak a finomhangolási pontosságokhoz képest, viszont limitált erőforrások is idő mellett ezek az eredmények is ígéretesek. Ha valaki egy több és nagyobb erőforrással rendelkező rendszeren próbálkozik meg ezzel, akkor több opciója és ideje van arra, hogy az AutoPrompt-al ennél még jobban és hatékonyabban megközelítse vagy akár meghaladja a finomhangolás során elért modellek teljesítményét.

Ami általánosságban elmondható a finomhangolási kísérletekről az az, hogy az adatok előzetes feldolgozása nem vezetett semmiféle javuláshoz, sőt inkább rosszabbul teljesítettek a „tisztított” bemeneteken tanuló modellek a „nyers” adatokat kapó modellekhez képest. Még egy ilyen megállapítás az, hogy a nagy modellek vizsgálata során a stabilizációs lépések elengedhetetlennek bizonyultak, ugyanis mindig sikerült a modellek pontosságát növelni, a validációs veszteséget pedig csökkenteni. A legjobb teljesítményt elérő modell a BERT alap variánsa lett, ami 0,9776-es pontosságot ért el 0,0941-es validációs veszteséggel.

Az AutoPrompt programmal elvégzett kísérletekről az mondható el, hogy legtöbb esetben az alapértelmezett tíz akkumulációs lépés, nyolcas kötegméretek, illetve az öt iterációs futások bizonyultak a legjobbnak a modellek pontosságának javításához. Ezek mellet még fontosnak bizonyult a megfelelő szavak megválasztása a címkéinkbe, amik közül a modell választhat, hogy megállapíthassuk annak pontosságát. A legjobb eredményt nyújtó modell az AutoPrompt segítségével a BERT nagy változata lett, mely 0,7821-es pontosságot ért el a kiértékeléshez használt adatkészleten.

Összességében az AutoPrompt ígéretes alternatívát nyújthat a hagyományosnak mondható finomhangolással szemben az álhírterjesztés elleni harcban: ahogy az előtanított nyelvi modellek egyre kifinomultabbak és hatékonyabbak lesznek ez a megközelítés potenciálisan fel is válthatja a finomhangolást. Viszont, ahogy az általam elért eredményekből is látható, egy erőforrás limitált környezetben jobban járunk a modellek finomhangolásával.

Irodalomjegyzék

1. Pekka Salokannel: The Impact of AI: How Artificial Intelligence is Transforming Society https://web.archive.org/web/20240215233008/https://www.3dbear.io/blog/the-impact-of-ai-how-artificial-intelligence-is-transforming-society [Utoljára hozzáférve: 2024-02-19]

2. Krystal Hu: ChatGPT sets record for fastest-growing user base - analyst note https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/ [Utoljára hozzáférve: 2024-02-19]

3. Pranshu Verma: The rise of AI fake news is creating a 'misinformation superspreader' https://www.washingtonpost.com/technology/2023/12/17/ai-fake-news-misinformation/ [Utoljára hozzáférve: 2024-02-19]

4. IBM: What is an AI model? https://www.ibm.com/topics/ai-model [Utoljára hozzáférve: 2024-02-19]

5. Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, Sameer Singh: AUTOPROMPT: Eliciting Knowledge from Language Models with Automatically Generated Prompts https://ucinlp.github.io/autoprompt/files/autoprompt-emnlp20.pdf [Utoljára hozzáférve: 2024-02-19]

6. Google: Colaboratory, Frequently Asked Questions https://research.google.com/colaboratory/faq.html [Utoljára hozzáférve: 2024-02-20]

7. Wikipédia: Hugging Face https://en.wikipedia.org/wiki/Hugging_Face [Utoljára hozzáférve: 2024-02-20]

8. Python Package Index: Transformers https://pypi.org/project/transformers/ [Utoljára hozzáférve: 2024-02-20]

9. Farkas Richárd: Gépi tanulás a gyakorlatban – Mély gépi tanulás (Deep learning) https://www.inf.u-szeged.hu/~rfarkas/ML20/deep_learning.html [Utoljára hozzáférve: 2024-02-20]

10. IBM: What is PyTorch? https://www.ibm.com/topics/pytorch [Utoljára hozzáférve: 2024-02-21]

11. Python Package Index: Datasets https://pypi.org/project/datasets/ [Utoljára hozzáférve: 2024-02-21]

12. Python Package Index: Evaluate https://pypi.org/project/evaluate/ [Utoljára hozzáférve: 2024-02-21]

13. Natasha Kassam: Disinformation and coronavirus https://www.lowyinstitute.org/the-interpreter/disinformation-coronavirus [Utoljára hozzáférve: 2024-02-22]

14. Parth Patwa, Shivam Sharma, Srinivas Pykl, Vineeth Guptha, Gitanjali Kumari, Md Shad Akhtar, Asif Ekbal, Amitava Das, Tanmoy Chakraborty: Fighting an Infodemic: COVID-19 Fake News Dataset https://huggingface.co/papers/2011.03327 [Utoljára hozzáférve: 2024-02-22]

15. Hugging Face: BERT https://huggingface.co/docs/transformers/model_doc/bert [Utoljára hozzáférve: 2024-02-23]

16. Anna Rogers, Olga Kovaleva, Anna Rumshisky: A Primer in BERTology: What We Know About How BERT Works https://aclanthology.org/2020.tacl-1.54.pdf [Utoljára hozzáférve: 2024-02-23]

17. Hugging Face: RoBERTa https://huggingface.co/docs/transformers/model_doc/roberta [Utoljára hozzáférve: 2024-02-23]

18. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov: RoBERTa: A Robustly Optimized BERT Pretraining Approach https://arxiv.org/pdf/1907.11692.pdf [Utoljára hozzáférve: 2024-02-23]

19. Hugging Face: DeBERTa https://huggingface.co/docs/transformers/model_doc/deberta [Utoljára hozzáférve: 2024-02-24]

20. Marius Mosbach, Maksym Andriushchenko, Dietrich Klakow: On The Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines https://openreview.net/pdf?id=nzpLWnVAyah [Utoljára hozzáférve: 2024-02-28]

Nyilatkozat

Alulírott Sprok Dániel, programtervező informatikus BSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszékén készítettem, programtervező informatikus BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat a Szegedi Tudományegyetem Diplomamunka Repozitóriumában tárolja.

2024. 04. 15.

Sprok Dániel

Köszönetnyilvánítás

A dolgozat az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében.