

Szegedi Tudományegyetem

Informatikai Intézet

**Számítógépes Algoritmusok és Mesterséges Intelligencia
Tanszék**

Nyelvtechnológiai csoport

**Nyelvi modellek teljesítményének összehasonlítása egy
kontextusalapú szójelentés-felismerés (Word-in-Context)
probléma megoldására és saját kiértékelő rendszer
fejlesztése Pythonban**

**Comparing the Performance of Language Models on a
Word-in-Context Benchmark and Developing a Custom
Evaluation System in Python**

Szakdolgozat

Készítette:

Fábián Bernát

Programtervező informatikus szakos
hallgató

Témavezető:

Dr. Berend Gábor Iván

egyetemi docens

Informatikai Intézet

**Számítógépes Algoritmusok és Mesterséges Intelligencia
Tanszék**

Nyelvtechnológiai csoport

Szeged

2025

Tartalomjegyzék

| | |
|--|-----------|
| Feladatkiírás | 7 |
| Tartalmi összefoglaló | 8 |
| Motiváció | 10 |
| Bevezetés | 11 |
| 1. Irodalmi áttekintés | 11 |
| 1.1. Az ambiguitás problémája a természetes nyelvekben | 11 |
| 1.2. A WiC probléma | 11 |
| 1.3. A WiC feladat és jelentősége | 12 |
| 1.4. Egy rendszer feladata a WiC adathalmazon | 12 |
| 1.5. A bináris osztályozás általában | 12 |
| 1.6. A WiC adathalmaz | 13 |
| 2. A szakdolgozat célja | 14 |
| 2.1. A kutatás célja | 14 |
| 2.2. A kiértékelő rendszer célja | 14 |
| 2.3. A függvénykönyvtár célja | 15 |
| 3. A megoldás módja: | 15 |
| 3.1. Eddigi legjobb megoldások | 16 |
| 4. Nagy nyelvi modellek | 17 |
| 4.1. <i>Mik a nagy nyelvi modellek?</i> | 17 |
| 4.2. <i>Mi a prompt?</i> | 17 |
| 4.3. Top-p és Top-k | 17 |
| 4.4. Google Colab Notebooks | 19 |

| | |
|--|-----------|
| 4.5. Chatbot Arena, vagy LMSYS - a kiértékelő platform | 19 |
| 4.6. Közösség rangsor | 19 |
| 4.7. A beadott szöveg eredete | 20 |
| 5. Az algoritmus válaszainak kiértékelése | 21 |
| 6. A beadott szöveg formátuma | 22 |
| 6.1. Egy mondat formátuma | 22 |
| 6.2. Egyszavas válaszok | 22 |
| 7. Konzisztencia | 26 |
| 7.1. OpenAI-modellek: a GPT család | 26 |
| 7.2. Anthropic Modellek: Claude család | 26 |
| 7.3. Microsoft Modellek: QWEN és Copilot család | 26 |
| 7.4. Google modellek: Gemini, Bard, Gemma | 26 |
| 7.5. DeepSeek-r1 és DeepSeek-V3 modell | 27 |
| 7.6. Twitter, xAI és Meta modellek | 27 |
| 7.7. Google AI Studio | 27 |
| 7.8. Egyéb említésre méltó modellek | 28 |
| 7.9. Megjegyzés | 28 |
| 7.10. Összegzés és következtetés | 28 |
| 8. Kérdés-válasz szótár | 28 |
| 8.0.1. 1.1.2.1 A beadott mondatok száma | 28 |
| 8.1. A nagy nyelvi modellek konfigurása a prompt beküldése előtt | 29 |
| 9. Kiértékelési metrikák | 29 |
| 9.1. Klasszifikációs módszer | 29 |
| 9.2. Saját annotáció készítése | 30 |
| 9.3. Megvizsgált modellek | 30 |

| | |
|---|-----------|
| 10. Saját rendszer fejlesztése | 34 |
| 10.1. Word2vec és BERT Elméleti háttér | 34 |
| 10.2. Tervezés | 34 |
| 10.2.1. Projekt tervezés fázis: | 34 |
| 10.2.2. A projekt felépítése | 35 |
| 10.3. Implementáció | 36 |
| 10.4. Tesztelés | 37 |
| 10.4.1. Tesztelés Benchmarking módszerrel | 37 |
| 10.4.2. Konzisztencia | 37 |
| 10.4.3. Egy példa kérdés: | 38 |
| 11. A 2 algoritmus eredményei és ábrák | 41 |
| 11.1. Modell teljesítmény értékelésének módszere | 41 |
| 11.2. A részletes kimutatásokat tartalmazó megoldás | 41 |
| 11.3. Kompaktabb megoldás | 43 |
| 12. Az NLTK Használata | 45 |
| 12.1. A WiC modell működése | 45 |
| 12.2. Adatfeldolgozás és annotáció | 45 |
| 12.3. Tanulságok, hipotézisek levonása | 45 |
| 12.4. Tervek a jövőre nézve. | 46 |
| 13. Összegzés | 46 |
| 14. Nyilatkozat | 47 |
| Nyilatkozat | 47 |
| Köszönetnyilvánítás | 48 |
| 15. Mellékletek | 49 |

| | |
|-------------------------------------|-----------|
| 16. Hivatkozások és források | 50 |
| Irodalomjegyzék | 51 |

Feladatkiírás

A hallgató feladata a nyelvtechnológia és azon belül a kontextusalapú szójelentés-felismerés problémakörének a megismerése, a Word-in-Context probléma megismerése, a jelenlegi legjobb megoldások áttekintése, összehasonlítása, valamint egy angol nyelvű WiC kiértékelő rendszer megtervezése, implementálása és tesztelése.

[leftmargin=2em, label=•]Ismerje meg az NLP és a nagy nyelvi modellek fogalomrendszerét. a gemma-2-2b-it modellen vizsgálja meg, hogy mennyire érzékeny a nyelvi modell arra, hogy a 2 mondatot fordítva kapja meg. Tehát az 'Ugyanazt jelenti-e a szó az "A" és "B" mondatban' eredménye ugyanaz, mint a 'Ugyanazt jelenti-e a szó a "B" és "A" mondatban' eredménye. Tervezze meg, implementálja és tesztelje a kontextusalapú szójelentés-felismerő rendszerét, a WiC: The Word-in-Context Dataset nyilvános és ingyenes Word-in-Context teljesítményteszt (benchmark) adathalmazra tervezve. Az adatok letöltése után végezze el a szükséges adattranszformációkat, az adattisztítási feladatokat, szűrje ki a felesleges, vagy használhatatlan adatok körét, illetve készítse el az elemzésre a kész adatállományt. Értékelje ki a kontextusalapú szójelentés-felismerő rendszerén a tesztalmozatot. Az elért eredményeiről készítsen statisztikai összesítéseket és látványos grafikus lekérdezéseket, ábrákat.

Elvégzett munkáit a Szakdolgozat keretében dokumentálja.

• Tartalmi összefoglaló

[leftmargin=2em, label=•] **A téma megnevezése** A Mohammad Taher Pilehvar által összeállított és publikált Word-in-Context (WiC) adathalmaznak a legmodernebb generatív nyelvi modelleken való kiértékelése, a gemma-2-2b-it modell konzisztenciájának a megvizsgálása, továbbá egy saját kontextusalapú szójelentés-felismerésre fejlesztett kiértékelő rendszer megtervezése, implementálása Python nyelven, majd tesztelése. Ez egy klasszifikációs probléma, tehát a Hallgató célja, hogy minél nagyobb pontosságot érjen el a célfeladatra. **A megadott feladat megfogalmazása** A feladat a kontextusalapú szójelentés-felismerés problémakörének a megismerése, jelenlegi legjobb megoldások áttekintése, valamint egy olyan angol nyelvű kontextusalapú szójelentés-felismerő rendszer implementálása és kiértékelése, amely képes legalább 60%-os pontosságot elérni a teszt adathalmazon. A rendszernek képesnek kell lennie a fedés és a precizitás értékét megadni, továbbá tévesztési mátrixot készíteni a predikciók eloszlásáról, az alábbi négy kategória szerint:

- **TP (True Positive):** Azok az esetek, amikor a modell helyesen ismeri fel, hogy a szó jelentése megegyezik a két mondatban.
 - **FP (False Positive):** Azok az esetek, amikor a modell tévesen gondolja azt, hogy a szó jelentése megegyezik, pedig valójában eltér.
 - **FN (False Negative):** Azok az esetek, amikor a modell nem ismeri fel a jelentésazonosságot, pedig a szó jelentése valójában megegyezik.
 - **TN (True Negative):** Azok az esetek, amikor a modell helyesen ismeri fel, hogy a szó jelentése különböző a két mondatban.
- **A megoldás lépései**
 - A jelenleg elérhető legnépszerűbb, legfejlettebb, úgy nevezett state-of-the-art" ingyenesen elérhető nagy nyelvi modellek megvizsgálása, többek között: Claude 3.7 Sonnet, Gemini-flash-thinking, Grok1, és a DeepSeek-r3.
 - Gemma-2-2b-it lokális telepítése, futtatása és kiértékelése
 - egyéb jó minőségű ingyenesen elérhető modellek kiértékelése az ollama alkalmazáson és a llmarena.ai webes platformon található modellek egy részén.
 - A saját implementáció megtervezése, implementálása és tesztelése
 - A kiértékelési módszerem bemutatása, hitelességének bizonyítása
 - **Alkalmazott technológiák, módszerek**
 - Az NLTK ¹ eszközcsoomag Lesk[?] eszköztárnak a Wordnet Pythonos implementációja a szinonimák behúzásához.
 - BERT szóbeágyazás építése a kétirányú mondatértelmezéshez és -összehasonlításhoz.
 - Gemma-2-2b-it, a gemma modell egy kvantált változata, a Hugging Face-ről letölthető

¹Natural Language Toolkit

- legkorszerűbb modellek az ollama felületről
- Gemini, GPT modellek, Claude Sonnet, Grok1, és a DeepSeek-v1 és DeepSeek-r3, QWEN-2.5, LMArena, LMSys, Nltk, Wornet, BERT, Gemma-2-2b-it, Huggingface, Transformers, PyTorch, Matplotlib.

- ***Elért eredmények***

Egyik elért eredményemnek a Codalab XL-WiC versenyeinek megfelelő kiértékelésem tartom, amely egy kifejezetten a WiC kiértékelő feladaton alapuló online verseny.

A másik nagy eredményem egy, a mintafeladathoz készült feldolgozó algoritmuscsomag, amely elvégzi a szükséges adattranzformációkat, az adattisztítási feladatokat és az adat értelmezhető formába alakítását.

Harmadik nagy büszkeségem, hogy 57.21%-os eredményt értem el a módszeremmel, ami közelít a WiC: The Word in Context dataset oldalon található eredményekhez.

- ***Kulcsszavak***

WiC, word-in-context, NLP, nyelvi modell, nagy nyelvi modell, LLM, szövegértelmező modell, információkinyerés, szövegkörnyezet, kontextus, mondatelemzés, szövektorok, számítógépes szemantika

Motiváció

Az informatika és a nyelvtechnika fejlődésével a generatív mesterséges intelligencia mindennapjaink részévé vált. A nagy nyelvi modellek kiértékelésére számos benchmark, azaz teljesítményteszt, metrika fejlődött ki az évek során. A legnépszerűbb ilyen benchmark a SuperGLUE Benchmark, amely 8, komoly kihívást jelentő feladat elé állítja a nagy nyelvi modelleket. Ebben a papírban a WiC feladatot mutatom be.

A WiC (Word-in-Context) probléma a SuperGlue 8 feladatának egyike. Az emberi szöveget értő nyelvi modellek fejlesztése kiemelten fontos mind az akadémiai kutatásban, mind a gyakorlati alkalmazásokban. Az angol nyelvű szövegek feldolgozása nem csak az angolszász területeken releváns, hanem Magyarországon is, hiszen az angol nyelv használata a számítógépek világában mindennapjaink része. Az egyetem és a helyi vállalatok is aktívan foglalkoznak természetesnyelv-feldolgozási (NLP) megoldásokkal. A munkahelyeken – különösen IT-területen – egy jól működő WiC modell jelentős előnyt biztosíthat kódok, dokumentumok, ügyfélkommunikáció vagy akár jogi szövegek értelmezésében, ami kulcsfontosságú lehet például a tudásközpontokban és startup cégeknél is. Egy hatékony WiC modell tehát nemcsak tudományos értékkel bír, hanem gyakorlati alkalmazásokban is közvetlen hasznot hozhat, főleg a nyelvészeti informatikai területen dolgozók számára.

Bevezetés

1. Irodalmi áttekintés

- TF-IDF és hasonlósági mérőszámok (baseline módszerek).
- WordNet, Lesk algoritmus és egyéb WSD technikák: Rövid magyarázat a működésről.
- Modern neurális módszerek: Transformer alapú modellek (pl. BERT).
- Ingyenes modellek használata Hugging Face és Ollama segítségével
- Benchmark adathalmazok és értékelési metrikák: pontosság (accuracy), fedés (recall), precizitás (precision), F1-score, tévesztési mátrix (confusion matrix).

A szóbeágyazások tervezésükből adódóan képtelenek modellezni a szavak szemantikájának dinamikus természetét, vagyis azt a tulajdonságot, hogy a szavak potenciálisan különböző jelentéseknek felelhetnek meg. E korlátozás kezelésére tucatnyi specializált jelentésreprezentációs technikát javasoltak, mint például a jelentés- vagy kontextualizált beágyazásokat. Azonban a téma kutatásának népszerűsége ellenére igen kevés olyan értékelési viszonyítási alap létezik, amely kifejezetten a szavak dinamikus szemantikájára összpontosít. Ebben a tanulmányban bemutatom, hogy a meglévő modellek túllépték az erre a célra szolgáló standard értékelési adatkészlet, azaz a Stanford Kontextuális Szóhasonlóság teljesítménykorlátját, és rámutatok annak hiányosságaira. A megfelelő viszonyítási alap hiányának kezelésére készült ez a nagyszabású, szakértők által összeállított annotációkon alapuló Word in Context (Szó a Kontextusban) adatkészlet, a WiC. Ezt javaslom a kontextusérzékeny reprezentációk általános értékelésére. A WiC elérhető a WiC: The Word-in-Context Dataset címen.

1.1. Az ambiguitás problémája a természetes nyelvekben

A természetes nyelvekben a programozási nyelvekkel ellentétben egy szónak több, egymástól teljesen elkülönülő jelentése is lehet. Például az "egér" szó jelenthet egy számítógépes perifériát vagy egy állatot, és a helyes értelmezéshez a környező szavakat ismerni kell. Az ilyen jellegű kétértelműségek automatikus feloldása az egyik központi problémája a természetes nyelvi rendszerek fejlesztésének.

1.2. A WiC probléma

Az emberi nyelvekben a többértelmű szavak attól függően, hogy milyen szöveggörnyezetben fordulnak elő, több, esetenként egymással össze nem függő dolgot is jelenthetnek. A "mainstream" statikus szóbeágyazások, mint például a Word2vec és a GloVe,

tervezésükből adódóan nem képesek modellezni a szavak szemantikájának dinamikus természetét, vagyis azt a tulajdonságot, hogy a szavak potenciálisan különböző jelentéseknek felelhetnek meg. Egy szóhoz mindig ugyanazt a szóvektort rendelik, kontextustól függetlenül. A kontextualizált szóbeágyazások kísérletet tesznek ennek a korlátnak a feloldására azáltal, hogy dinamikus reprezentációkat számítanak ki a szavakhoz, amelyek a szöveggörnyezet alapján képesek alkalmazkodni. Ilyen szóbeágyazás transzformer például a BERT, ám ennek is megvannak a korlátai.

1.3. A WiC feladat és jelentősége

A WiC feladat lényege, hogy egy adott szó két különböző mondatbeli előfordulásáról eldöntse, hogy azonos értelemben szerepel-e. A természetes nyelvi feldolgozásban ¹ ez kulcsfontosságú, mivel segít a szövegértelmezésben és a jelentésalapú keresés fejlesztésében.

1.4. Egy rendszer feladata a WiC adathalmazon

Egy rendszer feladata a WiC adathalmazon a szavak szándékozott jelentésének azonosítása. A WiC egy bináris osztályozási feladatként van megfogalmazva. Adott egy többjelentésű szó, amely mindkét mondatban előfordul, továbbá egy szófaj címke, 2 index és két szövegrészlet. Egy rendszer feladata, hogy meghatározza, hogy a szó ugyanabban a jelentésben használatos-e mindkét mondatban. A w célszó minden esetben csak egy ige, vagy főnév lehet. A célszóhoz két eltérő szöveggörnyezet tartozik. Ezen szöveggörnyezetek mindegyike a w egy specifikus jelentését váltja ki. A feladat annak megállapítása, hogy a w előfordulásai a két szöveggörnyezetben ugyanannak a jelentésnek felelnek-e meg vagy sem. Valójában az adathalmaz a gyakorlatban a szójelentés-egyértelműsítés (Word Sense Disambiguation) végrehajtásaként is értelmezhető.

1.5. A bináris osztályozás általában

TODO átfogalmazni A bináris osztályozási feladatok problémakörében új korszakba léptünk. Most már nem holmi egyszerű algoritmusok, reguláris nyelvtanok az elterjedt módszerek ha egy ilyen rendszert kell építeni. Helyette egyre inkább a neurális hálók és nagy nyelvi modellek az elterjedtek bináris osztályozó feladatokra. A Lesk-algoritmus[?] egy klasszikus szóértelmezés-felismerő módszer, amely a szótári definíciók és a kontextus összevetésével próbálja meghatározni a szó legmegfelelőbb jelentését. A legjobbak mégis a kifejezetten emberi szöveg megértésére specializálódott nagy nyelvi modellek, például a GPT-4.

¹Természetesnyelv-feldolgozás, számítógépes nyelvészet és nyelvtechnológia néven is hivatkoznak rá, angolul NLP

1.6. A WiC adathalmaz

A Word in Context dataset egy jó minőségű, nyelvészeti szakértők által készített benchmark adathalmaz. A mondatok a WordNetből, a VerbNetből és a Wikiszótárból származnak. A WiC adathalmaz segítségével megvizsgálhatjuk, hogy egy rendszer (modell, algoritmus) mennyire képes a szavak jelentését megérteni különböző kontextusokban. A WiC feladat része a SuperGlue Benchmarknak, amely 8 feladatból áll:

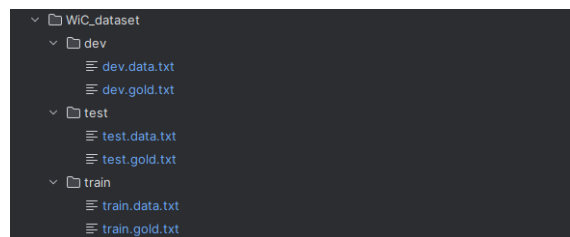
- BoolQ (Boolean Questions, Clark et al., 2019)
- CB (CommitmentBank, De Marneffe et al., 2019)
- COPA(Choice of Plausible Alternatives, Roemmele et al., 2011)
- MultiRC (Multi-Sentence Reading Comprehension, Khashabi et al., 2018)
- ReCoRD(Reading Comprehension with Commonsense Reasoning Dataset, Zhang et al., 2018)
- RTE(Recognizing Textual Entailment)
- **WiC(Word-in-Context, Pilehvar and Camacho-Collados, 2019)**
- WSC (Winograd Schema Challenge, Levesque et al., 2012)

A WiC tehát része a SuperGlue-nak, amely egy széles körben elfogadott benchmark nyelvi modellek kiértékelése körében. A SuperGLUE a GLUE továbbfejlesztett változata, amelyet 2019-ben vezettek be, mivel a GLUE feladatait a legmodernebb modellek (pl. BERT) már túl jól megoldották¹. Ez a legszélesebb körben elfogadott modell, amelyen többek között olyan intelligens modellek, mint a GPT-4o, Claude 3.7 Sonnet is megmérettetésre kerültek(TODO forrás: comparing GPT and Claude on benchmarks).

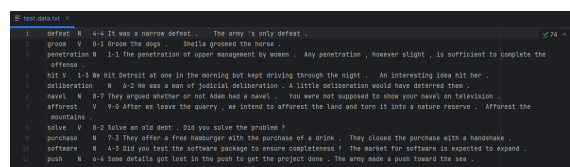
A WiC halmaz fel van osztva tanító, validációs és teszhalmazra, ezért neurális hálók tanítására is felhasználható. Ezt segíti elő az is, hogy az összes mondat tokenekre bontott, tabulált és egységesek az írásjelek is. A modelleket hasonlóan tanítják, mint ahogy az iskolában tanulnak a diákok. A benchmarkok feladatait jellemzően 3 részre vágják: tanító, validációs és teszhalmazra. Ennek az aránya eltérő lehet, de a tanítónak jóval nagyobbnak kell lennie, mint a másik kettőnek, és a teszhalmaznak nagyobbnak kell lennie, mint a validációs halmaznak. 80-10-10%-os eloszlás a standard, ezzel, vagy hasonló eredménnyel érhetőek általában el a legjobb eredmények.

A tanító adatbázist a korábbi iskolai példára visszatérve úgy képzelhetjük el, hogy ez a leadott anyag. A validációs halmaz olyan, mint egy mintavizsga, mint a ZH. A teszhalmaz pedig a végső megmérettetés, a vizsga. Fontos, hogy a teszhalmazon sose tanítsuk a modelleket, és sose legyen átfedés a halmazok között. Ez ugyanis magoláshoz vezet, és a modell nem lesz képes általánosítani.

¹Forrás: SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems



1. ábra. Az adathalmaz



2. ábra. Példa a tesztelő adat kinézetére

2. A szakdolgozat célja

2.1. A kutatás célja

A dolgozat célja elsősorban a kontextuális szóbeágyazások és a generatív nyelvi modellek mély megismerése, majd egy WiC kiértékelő rendszer létrehozása, amely különböző módszereket hasonlít össze, beleértve a hagyományos TF-IDF módszereket, Lesk algoritmust és WSD technikákat, valamint modern neurális megközelítéseket. A szakdolgozatom első felében azt vizsgálom meg, hogy népszerű nagy nyelvi modellek mennyire jól oldják meg a WiC problémát.

Ezután telepíték a gépemre egy nagy nyelvi modellt, melyet lokálisan tesztelek és konfigurálok, minél jobb eredmény elérése érdekében.

Végül saját modellt hozok létre, amely képes legalább 60%-os pontosságot elérni a teszt adathalmazon, emellett a fedést (recall), a precizitást (precision) és az F1-score-t megadni, tévesztési mátrixot készíteni a TP, FP, FN, TN címkék eloszlásáról. Ennek a részleteiről a Nagy Nyelvi Modellek Eredményei fejezetben írok.

2.2. A kiértékelő rendszer célja

A készített alkalmazás célja, hogy a kiértékelő algoritmus minél nagyobb pontosságot érjen el a WiC - The Word in Context Dataset tesztadathalmazon. A szakdolgozatom a **WiC Decision Maker Model** alkalmazására és optimalizálására összpontosít, amely egy Python alapú alkalmazásként valósul meg. A projekt célja egy olyan rendszer fejlesztése,

amely a Word-in-Context (WiC) feladatokhoz nyújt támogatást, lehetővé téve a szavak kontextusbeli jelentésének hatékony felismerését és osztályozását.

2.3. A függvénykönyvtár célja

A kiértékelő rendszerem mellé implementált Python algoritmus- és függvénykönyvtáram egy önálló, standalone modulként, vagy package-ként is használható, amelyet elérhetővé tervezek tenni bárki számára az interneten.

A Python 3.12 virtuális környezete lehetőséget biztosít modellek egyszerű integrációjára a rendszerembe. Az alkalmazás magában foglalja a gépi tanulási modellt, amely a WiC adathalmaz feldolgozására és azonosítására épül. A rendszer teljesítményének növelése érdekében különböző optimalizációs stratégiákat és kvantált modelleket is vizsgálok.

A projekt további célkitűzései közé tartozik a felhasználóbarát interfész kialakítása, amely lehetővé teszi a tesztesetek futtatását és az eredmények vizualizációját. Az alkalmazás fejlesztése során olyan technológiákat és eszközöket használok, mint a Hugging Face Transformers, PyTorch és a SQLite adatbázis-kezelő, hogy biztosítsam a rendszer hatékonyságát és skálázhatóságát.

Ennek megvalósítása érdekében az alábbi platformokat vizsgálom meg:

- Ollama
- vLLM projekt
- Hugging Face Transformers
- Llama.cpp

3. A megoldás módja:

- A szakdolgozatom első felében elemeztem a már meglévő módszereket, megpróbáltam minél jobb ingyenes nyelvi modellt találni, azt úgy paraméterezni és konfigurálni, hogy a WiC feladatot minél általánosabban, minél jobb eredménnyel meg tudja oldani.
- Összehasonlítottam a webes és a lokális nyelvi modell futtatás előnyeit és hátrányait.

State-of-the-Art

| Sentence-level contextualised embeddings | Implementation | Accuracy % |
|--|---------------------------|-------------|
| SenseBERT-large† | Levine et al (2019) | 72.1 |
| KnowBERT-W+W† | Peters et al (2019) | 70.9 |
| RoBERTa | Liu et al (2019) | 69.9 |
| BERT-large | Wang et al (2019) | 69.6 |
| Ensemble | Gari Soler et al (2019) | 66.7 |
| ELMo-weighted | Ansell et al (2019) | 61.2 |
| Word-level contextualised embeddings | Implementation | Accuracy % |
| WSD† | Loureiro and Jorge (2019) | 67.7 |
| BERT-large | WIC's paper | 65.5 |
| Context2vec | WIC's paper | 59.3 |
| Elmo | WIC's paper | 57.7 |
| Sense representations | | |
| LessLex | Colla et al (2020) | 59.2 |
| DeConf | WIC's paper | 58.7 |
| SW2V | WIC's paper | 58.1 |
| JBT | WIC's paper | 53.6 |
| Sentence level baselines | | |
| Sentence Bag-of-words | WIC's paper | 58.7 |
| Sentence LSTM | WIC's paper | 53.1 |
| Random baseline | | 50.0 |

† Use external knowledge resources

3. ábra. Ez a táblázat különböző nyelvi modelleket rangsorol a WiC feladaton elért pontosságuk alapján.

A képen látható táblázat magyarázata

A WiC feladat bemutatása A WiC feladat annak eldöntését kéri, hogy egy célszó ugyanazt a jelentést hordozza-e két különböző szöveggörnyezetben. Ez egy összetett NLP probléma, mivel ötvözi a szójelentés-egyértelműsítés (WSD) és a kontextuális beágyazások elemeit.

- A Peternity egy konzolos és webes alkalmazás, amelynek célja a WiC (Word in Context) feladatok megoldásának vizsgálata különböző nyelvi modellek segítségével. Az alkalmazás arra is képes, hogy elemezze, mennyire érzékenyek ezek a modellek arra, ha a két példamondat sorrendjét felcseréljük.
- A program fejlesztését inkrementális modellel végeztem, tehát kezdetben csak vázlatosan határoztam meg a kiértékelő rendszer funkcióit. Ennek a modellnek az előnye például a vízesés modellel szemben az, hogy a követelmények nincsenek konkrétan lerögzítve a fejlesztés elején, hanem folyamatosan lehet rajtuk változtatni. Ha eszembe jut egy újabb módszer, amivel jobb eredményt lehet elérni, vagy egy funkció, amely növeli a program érthetőségét, átláthatóságát, akkor nem kell újratervezni a programot.

3.1. Eddigi legjobb megoldások

4. Nagy nyelvi modellek

4.1. Mik a nagy nyelvi modellek?

A nagy nyelvi modell (angolul Large Language Model, LLM) olyan számítási modell, amely képes nyelv generálására vagy más természetes nyelvi feldolgozási feladatok elvégzésére. Ezek a modellek hatalmas mennyiségű szöveges adat feldolgozásával és mélytanulási technikák alkalmazásával sajátítják el a nyelv megértését és előállítását. Az LLM-ek, mint például az OpenAI GPT-sorozata, a Google PaLM vagy a Meta LLaMA modelljei, különböző architektúrákat használnak, de leggyakrabban a transzformer alapú megközelítést alkalmazzák. Mint nyelvi modellek, az LLM-ek úgy sajátítják el ezeket a képességeket, hogy óriási mennyiségű szövegből, egy önfelügyelt és egy félig felügyelt tanulási folyamat során, statisztikai összefüggéseket tanulnak meg. Ezek a modellek képesek különféle feladatok elvégzésére, mint például szövegfordítás, összefoglalás, kérdés-válasz és kreatív szövegalkotás. Azonban fontos megjegyezni, hogy az LLM-ek teljesítménye és alkalmazhatósága nagymértékben függ a betanításukhoz használt adatok minőségétől és mennyiségétől.

4.2. Mi a prompt?

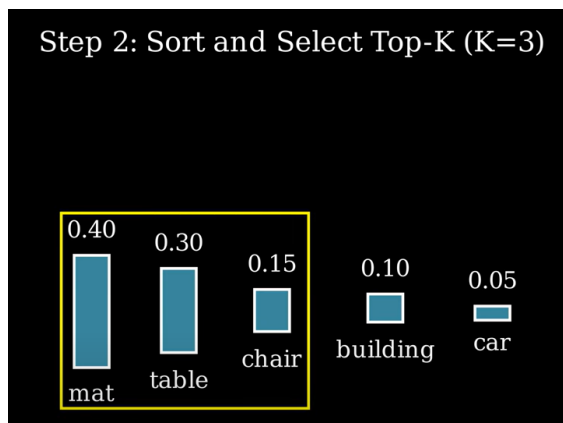
A prompt az informatikában egy nagy nyelvi modell számára beadott input, amelyre a modell egy előre meghatározott kimenetet generál. Működési elve lényegében ugyanaz, mint egy hagyományos konzolnak, parancssori interfésznek, például egy Windows rendszeren a PowerShell, vagy Linuxon Bash terminállal, azonban prompt esetén a kommunikáció emberi nyelven történik. Egy jelentős előnye a promptolásnak, hogy nem csak egy előre meghatározott parancskészletet használhatunk, hanem bármit beírhatunk, nincsenek szintaktikai hibák vagy futtatási hibák („error”), mivel a rendszer minden bemenetre képes válasz generálására. Azonban, mivel cloud alapú rendszerek, hálózati hiba előfordulhat. Hátránya a cloud alapú AI chatbot szolgáltatásoknak még, hogy nagyon erőforrásigényesek, emiatt rengeteg CO2 kibocsájtást vonnak maguk után.

4.3. Top-p és Top-k

[H] Tegyük fel, hogy a nyelvi modellnek a következő mondatot kell befejeznie:

The cat sat on the _ .

A modell célja, hogy megtalálja a legmegfelelőbb szót a hiányzó helyre. Ehhez különböző mintavételi stratégiákat alkalmazhat, például a Top-k vagy Top-p (más néven nucleus sampling) eljárást. A Top-k módszer során a modell kiszámítja az összes lehetséges folytatás valószínűségét, majd ezek közül kiválasztja a k legvalószínűbbet, a többit pedig elveti. A kiválasztott k szóból ezt követően véletlenszerűen választ egyet, amelyet a szöveg



5. ábra. Enter Caption

folytatásához használ. Ez a megközelítés biztosítja, hogy csak a legrelevánsabb szavak kerüljenek figyelembe vételre, ugyanakkor némi változatosságot is megtart a generálásban. A példában az alábbi szöveget kell folytatnia:

Input Phrase: 'The cat sat on the'

4. ábra. Top k input sentence

Ehhez a modell sorba rak tetszőleges számú odaillő folytatási szót, a valószínűségük alapján: Tehát a *Top k* esetén az algoritmus kiválaszt belőlük pontosan 'k' a legvalószínűbb 'kdb szót, a többit eldobja. A megmaradt szavakból random választ egyet.

A top 'p' működése hasonló. Annyiban tér el, hogy nem egy fix számú legvalószínűbb szót választunk ki, hanem az odaillő szavak valószínűsége szerint. A 'p' egy 0.00-tól 1.00-ig terjedő valószínűség. A szavakat akkumulatíván vesszük számításba, amíg a p érték nagyobb, mint a soron következő legvalószínűbb szó valószínűsége, addig bele vesszük a szót és ugrunk előre. Ellenkező esetben csak bele vesszük a szót, és leáll az algoritmus. Ezzel a módszerrel minél nagyobb a p érték, annál több szó közül lehet választani, de legalább 1 szót kapunk, tehát mindenképpen tudjuk folytatni a szöveget.

Miért használtam őket a feladat megoldásához? Saját, a nagy IT cégek modelljeihez hasonló méretű és minőségű szoftver készítése egyéni programozóként gyakorlatilag lehetetlen, sok embert és hatalmas mennyiségű, jó minőségű adatot igényel. Ilyen nagy mennyiségű adat tárolásához és feldolgozásához sem kapacitásom, sem hardveres erőforrásaim nem voltak. Helyette a Google Colab Notebook webes Python futtatókörnyezetet, Hugging Face modelleket, és a nagy cégek modelljeinek webes API-jait használtam. A nagy cégek nagy nyelvi modelljeinek előnye, hogy hatalmas adathalmazban vannak tanítva, például az összes New York Times rovat fél évszázadra visszamenőleg, a teljes Wikipédia szövege, és rendszerint egy közösségi média oldal (például a Llama modellek a Meta közösségi média oldalaiból tanult rengeteget, a Grok-ot az X adatain tanítják, a Google pedig felhasználja többek között a kereső, Gmail, Drive, fotók szolgáltatásainak felhasználói adatait a modelljei fejlesztéséhez). Szinte mindenre láttak már példát - talán a dolgozatom alapját szolgáló Pilehvar által kiadott WiC adathalmazt is látták már, hiszen része a SuperGlue benchmarknak, mellyel számos modellt kiértékelnek (de lehet, hogy tanítják is a train adathalmazzal).

4.4. Google Colab Notebooks

Ahhoz, hogy nagy nyelvi modell válaszait emberi időben megkapjuk, anélkül, hogy órákig felhasználnánk a személyi gépünk teljes erőforrását, GPU-n kell futtatni a modellt. Ez megtizedeli a futási időt. Az általam elérhető eszközök egyike sem GPU-s. Erre kínált megoldást a Google Colab. Ez a platform ugyanis kínál GPU-s futtatókörnyezetet. Továbbá az is előnye, hogy nem a gép háttértárát, memóriáját stb. használja, hanem egy távoli gépét, amit a Google biztosít.

4.5. Chatbot Arena, vagy LMSYS - a kiértékelő platform

A különböző nagy nyelvi modellek összehasonlítására a <https://lmarena.ai/> platformot használtam. Ez az oldal minden ismertebb ingyenes nagy nyelvi modellhez API-n keresztüli limitált hozzáférést biztosít. 2025. április 20-án 171 modell API-ja volt meghívható az oldalon keresztül. Ezek közül a leginkább említésre méltóak a GPT-3.5, GPT-4o, GPT-4o mini, Gemini-flash, Gemini-pro, Gemini-flash-pro-thinking, Llama, grok, DeepSeek-r1 és r3, Claude Sonnet, Claude Haiku. Az elérhető modellek listája folyamatosan változik.

A LMarena 2025 áprilisában céget is alapított.

4.6. Közösség rangsor

Az oldal több mint 1.5 millió ember szavazatán alapszik. A "ranking" dropdown menüben az opciók közül az "English" kategóriát választottam, mert a problémám ehhez a területhez (domainhez) áll a legközelebb. A közösség által a következő sorrendben voltak rangsorolva a nagy nyelvi modellek (2025.02.25-én, a teszt időpontjában):

The screenshot shows the Chatbot Arena LLM Leaderboard. At the top, it says 'Chatbot Arena LLM Leaderboard: Community-driven Evaluation for Best LLM and AI chatbots'. Below this, there's a table with columns: Rank, Delta, Model, Arena Score, Win % (vs GPT-4), Loss %, Organization, and License. The table lists several models, with GPT-4 Turbo (OpenAI) at the top.

| Rank | Delta | Model | Arena Score | Win % (vs GPT-4) | Loss % | Organization | License |
|------|-------|---|-------------|------------------|--------|--------------|-------------|
| 1 | | GPT-4 Turbo (OpenAI) | 1.000 | 100% | 0% | OpenAI | Proprietary |
| 2 | | Gemini 1.5 Flash (Google) | 0.985 | 98.5% | 1.5% | Google | Proprietary |
| 3 | | Gemini 1.5 Pro (Google) | 0.975 | 97.5% | 2.5% | Google | Proprietary |
| 4 | | Gemini 1.0 Pro (Google) | 0.965 | 96.5% | 3.5% | Google | Proprietary |
| 5 | | Gemini 1.0 Pro Vision (Google) | 0.955 | 95.5% | 4.5% | Google | Proprietary |
| 6 | | Gemini 1.0 Pro Ultra (Google) | 0.945 | 94.5% | 5.5% | Google | Proprietary |
| 7 | | Gemini 1.0 Pro Experimental (Google) | 0.935 | 93.5% | 6.5% | Google | Proprietary |
| 8 | | Gemini 1.0 Pro Experimental Vision (Google) | 0.925 | 92.5% | 7.5% | Google | Proprietary |
| 9 | | Gemini 1.0 Pro Experimental Ultra (Google) | 0.915 | 91.5% | 8.5% | Google | Proprietary |
| 10 | | Gemini 1.0 Pro Experimental Ultra Vision (Google) | 0.905 | 90.5% | 9.5% | Google | Proprietary |

6. ábra. Llm-ek rangsora

4.7. A beadott szöveg eredete

A nagy nyelvi modellek kiértékelésénél nagyon nem mindegy, hogy honnan vesszük a kiértékelő adathalmazt. A https://pilehvar.github.io/wic/package/WiC_dataset.zip megfelelő értékelési referenciaadatkészlet, hiszen szakértők által gondosan annotált példákon alapul. Mivel a nagy nyelvi modellek már be vannak tanítva, és validálva vannak, ezért a kiértékelésükhöz a tanító (train) és a validációs (dev) adathalmazokra nem volt szükség, csak a teszt (test) halmazra.

Az WiC adathalmazban található mondatok tokenizálva vannak, azaz önálló tokenként tekintenek a következőkre is:

1. vessző,
2. mondatvégi pont,
3. az angol birtokos személyjel *'s* rag,
4. a *n't* tagadószócska.
5. *'ve*, *'d*, *'ll*, *'re*, *'m* és bármilyen aposztrófot használó angol kifejezés

Ezek szóközzel el vannak választva az előttük és utánuk található tokentől. Ezekre külön figyelniem kellett, amikor természetes nyelvezetűvé alakítottam a szöveget.

Emellett volt pár kifejezés és speciális karakter, amelyet Pythonban escape-elni kellett. A mondatokat ugyanis string dictionary-ként tároltam el, amelyekben a vessző (') és kettőspont (:) karaktereknek speciális funkciója van. Így az alábbi helyzetekben gondosan átalakítottam a szöveget Python számára parse-olható alakba.

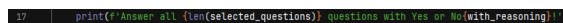
Példák:

1. My boss is out on another of his three martini lunches . Will you join us at six o'clock for martinis ? -> My boss is out on another of his three martini lunches. Will you join us at six o'clock for martinis?



```
def make_sentence_human_readable(sentence):
    """Replaces contractions for better readability both for humans and for chatbots."""
    sentence = sentence.replace("isn't", "is not")
    sentence = sentence.replace("it's", "it is")
    sentence = sentence.replace("n't", "not")
    return sentence
```

7. ábra. C:\PycharmProjects\Peternity\modules\wic_sentence_normalizer.py



8. ábra. A nagy nyelvi modellek promptját úgy kezdtem, hogy érveljen minden mondat esetében

2. The derivative of $f : f(x) = x^2$ is $f' : f'(x) = 2x$. 'electricity' is a derivative of 'electric'. -> The derivative of $f : f(x) = x^2$ is $f' : f'(x) = 2x$. 'electricity' is a derivative of 'electric'.

Ez azt jelentette, hogy ezek a tokenek szóközzel el vannak választva a megelőző és következő tokenről. Én azonban a kutatómunkám első fázisaként a weben, távoli szerverten futó modelleken szerettem volna tesztelni, ahova a mondatok ilyen formában nem feleltek meg a kiértékelésre, mert pont az volt a célom, hogy megvizsgáljam, mennyire értik meg jól a nagy nyelvi modellek az emberi szöveget. Emiatt a listában felsorolt tokeneknek az előfordulásainál átalakítottam a szöveget: kitöröltem az előttük lévő szóközt. Ezzel lényegében visszaalakítottam a mondatokat emberi nyelvre. Erre az alábbi szkriptet használtam:

5. Az algoritmus válaszainak kiértékelése

Hamar be kellett látnom, hogy az algoritmus egy pusztán "Yes" vagy "No" válasza önmagában nem sok információt rejt magában. Ha csak ennyit kapok vissza, nem tudom, hogy hogyan gondolkodott a modell, milyen szóvektori vannak, mekkora volt a context window-ja, mennyi volt a top p, top k, temperature, stb. Ezek mind nagyon hasznos információkat tartalmaznak mi, fejlesztők számára. Emiatt, hogy a pusztán "Yes" és "No" válasznál többet kapjak vissza, 2 módszert alkalmaztam:

1. A nagy nyelvi modellek promptját úgy kezdtem, hogy érveljen minden mondat esetében:

```
f'Answer all {len(selected_questions)} questions with Yes or No
```

- . Tapasztalatom szerint ez elég volt ahhoz, hogy részletes válaszokat kapjak a modellek webes futtatása esetében, de lokális futtatásnál is segített tájékozódni.
2. 0.00-tól 1.00-ig terjedő skálán, azaz egy *normalizált skálán* ábrázoltam, hogy a modell mennyire biztos abban, hogy a célszó a 2 mondatban ugyanabban az értelemben van jelen. 1.00 jelenti azt, hogy a modell teljesen biztos abban, hogy a célszó a 2 mondatban ugyanazt jelenti, 0.00 esetén pedig teljesen biztos az ellenkezőjében.

6. A beadott szöveg formátuma

6.1. Egy mondat formátuma

A 'promptszótár formátuma az alábbi:

```
'Does the word "defeat" mean the same thing in sentences "It was a nar
'Does the word "groom" mean the same thing in sentences "Groom the dog
'Does the word "penetration" mean the same thing in sentences "Th
```

6.2. Egyszavas válaszok

A szótárból előállított prompt formátuma pedig a következő:

```
Answer all `n` questions with Yes or No!
Does the word "defeat" mean the same thing in sentences "It was a nar
Does the word "groom" mean the same thing in sentences "Groom the dog
.
.
.
(n sor összesen)
```

A modellek erre a formátumra szinte mindig csak egy 'Yes'vagy Noszóval válaszolnak, ezt saját teszteléssel is megerősítettem. Ezeket az előrejelzéseket egy egyszerű leképezéssel össze lehet hasonlítani a gold fájlokban található T'(True) vagy 'F'(False) címkékkel. A modell válaszána a gold standarddal ² való összevetésekor minden esetet az alábbi négy kategória egyikébe sorolhatjuk:

- **True Positive (TP):** A modell helyesen válaszol 'Yes-t, amikor a gold címke T', azaz valóban azonos a szó jelentése.
- **False Positive (FP):** A modell tévesen válaszol 'Yes-t, miközben a gold címke 'F', tehát eltér a szó jelentése.
- **False Negative (FN):** A modell tévesen válaszol No-t, miközben a gold címke T', tehát azonos lenne a szó jelentése.
- **True Negative (TN):** A modell helyesen válaszol No-t, amikor a gold címke 'F', azaz valóban különböző a szó jelentése.

Eltérő kimenetet kapok, ha érvelés kérésével adom be a modelleknek a szöveget:

²A "gold standard" értékek nem más, mint nagy mennyiségű, szakértők által annotált, hitelesített példa-feladat, megoldásokkal.

Answer all `n` questions with Yes or No with reasoning!

Does the word "defeat" mean the same thing in sentences "It was a nar

Does the word "groom" mean the same thing in sentences "Groom the dog

.

.

.

(n sor összesen)

Ebben az esetben a chatbotok 1-2 mondatban meg szokták magyarázni a döntésük mögötti logikai érvelést. Ez is hasznos információ, amelyet felhasználok a modell teljesítményének megítéléséhez.

A modellek válaszát egy Google Sheets fájlba mentettem el, az alábbi struktúrában:

| Word | Gold | gemini-2.0 | gemini-2.0 | Claude 3.7 Sonnet | Claude 3.7 | gemini-2.5 |
|--------------|------|-------------------|--------------------|-----------------------|-------------------|--------------|
| defeat | T | Yes. In both ser | Yes. Both refer | defeat: Yes. In both | No - "defeat" as | 1. **defeat |
| groom | T | Yes. In both ser | Yes. Both refer | groom: Yes. Both | No - "groomed" | 2. **groom |
| penetration | T | No. "Penetration" | No. The first is j | penetration: No. Th | No - physical pe | 3. **penetr |
| hit | F | No. The first "hi | No. The first is i | hit: No. "Hit Detroit | No - an idea occ | 4. **hit++ |
| deliberation | F | Yes. In both ser | Yes. Both refer | deliberation: Yes. I | Yes - both refer | 5. **deliber |
| navel | T | Yes. In both ser | Yes. Both refer | navel: Yes. Both re | Yes - both refer | 6. **navel+ |
| afforest | T | Yes. In both ser | Yes. Both refer | afforest: Yes. Both | Yes - both refer | 7. **affore+ |
| solve | F | No. The first "sc | No. The first is i | solve: No. "Solve a | No - solving a p | 8. **solver |
| purchase | T | No. The first "pu | No. first is the a | purchase: No. The | Yes - both refer | 9. **purch |
| software | T | Yes. In both ser | Yes. Both refer | software: Yes. Both | Yes - both refer | 10. **softw |
| push | T | Yes. In both ser | No. The first is i | push: Yes. Both ser | Yes - both refer | 11. **push++ |
| bake | T | No. The first ser | Yes. Both refer | bake: No. In "potat | Yes - both refer | 12. **bake++ |
| relieve | F | Yes. In both ser | Yes. Both refer | relieve: Yes. Both s | Yes - both refer | 13. **reliev |
| style | T | Yes. In both ser | Yes. Both refer | style: Yes. Both ser | Yes - both refer | 14. **style+ |
| crumb | F | No. The first "cr | No. The first is i | crumb: Yes. Both | No - adding bre | 15. **crumb+ |
| include | F | Yes. In both ser | Yes. Both refer | include: Yes. Both | Yes - both refer | 16. **includ |
| companion | T | Yes. In both ser | Yes. Both refer | companion: Yes. B | Yes - both refer | 17. **compar |
| reveal | T | No. The first "re | No. First is to d | reveal: No. The fir | No - revealed f | 18. **reveal |
| presence | F | No. The first "pr | No. first is phys | presence: Yes. Bot | Yes - both refer | 19. **presen |
| relax | T | No. The first "re | No. The first ref | relax: No. "Don't rel | No - rules beco | 20. **relax+ |
| parity | F | No. The first "pa | No. The first is i | parity: No. The fir | No - parity refer | 21. **pariti |
| raise | T | Yes. In both ser | Yes. Both refer | raise: Yes. Both ser | Yes - both refer | 22. **rais+ |
| suspend | F | No. The first me | No. The first is j | suspend: No. "Sun | Yes - both refer | 23. **suspen |
| amass | T | Yes. In both ser | Yes. Both refer | amass: Yes. Both | Yes - both refer | 24. **amass+ |
| term | F | No. The first "te | No. The first is i | term: No. "Full term | No - vocabulary | 25. **term++ |
| leash | T | No. The first me | No. The first is i | leash: Yes. Both ser | Yes - both refer | 26. **leash+ |
| conversion | F | No. The first "cc | No. The first is i | conversion: No. Th | No - religious co | 27. **conver |

9. ábra. A tesztadatok és rajtuk végzett predikciók Google Sheets-ben

| Word | Gold | gemini-2.0 | gemini-2.0 | Claude 3.7 Sonnet | Claude 3.7 | gemini-2.5 |
|-----------------|------|------------|------------|-------------------|------------|----------------|
| holder | T | | | | | 1379. **hold |
| chapter | F | | | | | 1380. **chap |
| put | F | | | | | 1381. **put+ |
| deflate | F | | | | | 1382. **deflat |
| love | T | | | | | 1383. **love+ |
| feed | F | | | | | 1384. **feed+ |
| age | T | | | | | 1385. **age+ |
| counsel | T | | | | | 1386. **coun+ |
| leak | F | | | | | 1387. **leak+ |
| repair | T | | | | | |
| wall | T | | | | | |
| agree | T | | | | | |
| place | F | | | | | |
| nursing | T | | | | | |
| naught | F | | | | | |
| onrush | T | | | | | |
| correction | F | | | | | |
| fumble | F | | | | | |
| extent | T | | | | | |
| stress | F | | | | | |
| build | T | | | | | |
| exchange | T | | | | | |
| RESULTS | | | | | | |
| Total True | | 798 | 34 | 45 | 36 | 47 |
| Total T+F | | 1498 | 69 | 79 | 60 | 79 |
| True/All % | | 50.00% | 49.28% | 56.96% | 60.00% | 59.49% |
| Match with Gold | | | 72.46% | 67.09% | 70.00% | 72.15% |

10. ábra. A predikciók összesítése és pontosság számolás Google Sheets-ben

| | Gold | gemini 2.0 | gemini 2.0 | Claude 3.7 Son | Claude 3.7 Son | gemini 2.5 | gemini 2.5 | Column 3 | Column 4 |
|---------|------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| t | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| n | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| ization | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| ation | F | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| ost | F | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| ase | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| are | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| e | F | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| b | F | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| de | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| anion | T | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| nce | F | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is | No. The first is |
| | | | | | | | | | |

11. ábra. Reguláris kifejezések használata a válaszok kinyeréséhez

Az első sorban szerepelnek a gold standard értékek. A TP, FP, FN, TN címkék számát és eloszlását a Google Táblázatok beépített függvényei segítségével határoztam meg, és szimplán összeszámoltam a modellek által predikált "Yes" és "No" válaszok soronkénti megegyezését az azonos sorban szereplő gold standard értékekkel.

Egyes modelleknél (pl. Gemini 2.5 Pro Experimental 03-25) problémát okozott az, hogy nem csupán egy 'Yes', vagy egy 'No' válasszal tért vissza, hanem gondolatmenetet, magyarázatot is hozzáfűzött - akkor is, ha a promptban ennek az elhagyására kértem. Így pl. az "I know! The answer is Yes." szöveg a Google Táblázatok függvényemben 'No'-ként lenne beleszámítva, akkor is, ha utána a modell szerint. Emiatt ezeknek a modellek outputjának a feldolgozásakor reguláris kifejezést (regexet) kellett használnom (x.y ábra).

7. Konzisztencia

A következő fejezetekben arra válaszolok, hogy a különböző nyelvi modellek mennyire érzékenyek arra, hogyha a 2 példamondatot, amire vonatkozóan döntést kell hozniuk, azt fordított sorrendben kapják meg. A modelleket fejlesztőik alapján szedtem össze és csoportosítom.

7.1. OpenAI-modellek: a GPT család

A GPT modellek transzformátor architektúrára épülnek, és az inputban a szavak pozícióját is figyelembe veszik. A mondatok sorrendjének felcserélése befolyásolja az eredményt, ám leginkább csak akkor, ha a két mondat tartalma eltérő kontextust biztosít a vizsgált szó számára. Ez a WiC adathalmazban ritkán fordul elő, ám tapasztalataim szerint előfordulnak következetlen válaszok.

7.2. Anthropic Modellek: Claude család

- Korábbi verziók: Claude 3 Haiku, Claude 3 Sonnet, Claude 3 Opus, Claude 3.5 Sonnet
- Legújabb, *state-of-the-art* verziók: Claude 3.7 Haiku, Sonnet és Opus: A Claude modellek pontos architektúráis részletek kevésbé nyilvánosak. Ezeknek a modelleknek a fő erőssége a kódolásban való kiváló segítségnyújtás, de párbeszédkonzisztencia és érvelési következtetés szempontjából kiemelkedőek. Feltételezhető, hogy a bemeneti mondatok sorrendjének megváltoztatása hatással lehet a modell által generált válaszra.

7.3. Microsoft Modellek: QWEN és Copilot család

A Copilot főként kódkiegészítésre optimalizált, így a nyelvi kontextus finomságai, mint a szójelentéskülönbség, nem a fő erősségei – emiatt a sorrendváltozásra adott válaszok is ingadoznak.

7.4. Google modellek: Gemini, Bard, Gemma

A Google modelljei igen előkelő helyet foglalnak el a <https://lmarena.ai/> rangsorán, a Gemini-2.5-Pro-Exp-03-25 modellje az első helyet foglalja el, és a rangsor első 16 modelljéből 7 Google által fejlesztett³. A gyakorlatban sokszor félreértelmezik a felhasználó

³20205.04.23-án

szándékát, de a WiC-feladatra nagyon jól teljesítenek. A teljes teszt halmazon 80%-os pontosságot ért el, egyenes és fordított sorrendben is megadva.

7.5. DeepSeek-r1 és DeepSeek-V3 modell

A WiC (Word-in-Context) feladatokon a modell gyakran eltérő eredményt ad, ha a példamondatok sorrendje megváltozik. Bár 128 ezer tokenes kontextusablakot használnak, a modell gyakran "elveszti" a korábbi információkat, ha a bemenet hosszú.

7.6. Twitter, xAI és Meta modellek

- Grok
 - Grok 3 Beta x A Grok-3 Beta modellje erősen marketingelt, de a gyakorlatban túlértékeltnek bizonyul:
Aggresszív, de pontatlan: A modell gyakran túlbizonyos, hibás válaszokat ad önbizalommal, és nehezen ismeri be a tévedéseit.
Politikai elfogultság: Elon Musk nyilvános állításainak megfelelően a modell tendenciális lehet bizonyos társadalmi kérdésekben, ami a semlegességét kérdőjelezi meg.
Tesztelési hiányosságok: Az xAI nem tett közzé átfogó benchmark eredményeket a Grok-ról, így nehéz összehasonlítani más modellekkel.
- Llama
 - Llama 3.2 kipróbálható Ollama telepítés után, konzolon keresztül Hátrányok:
Közepes konzisztencia: A mondatrend-változásra kevésbé érzékeny, mint a DeepSeek, de a válaszok minősége még mindig ingadozó.
Korlátozott kreativitás: A kreatív feladatokban (pl. történetírás, metaforaértelmezés) gyakran sablonos válaszokat ad.
Telepítési nehézségek: Míg a GPT-4 vagy Claude könnyen elérhető felhőalapú szolgáltatáson keresztül, a Llama helyi futtatása (pl. Ollama-val) nehézkes, vagy egyenesen lehetetlen, komoly technikai akadályokba ütközhetünk.

7.7. Google AI Studio

Hosszas internetes böngészés után rátaláltam a Google AI Studiora. Ez ingyenes, Google által fejlesztett platform, ahol megtalálható az összes 'state-of-the-art' modelljük, és rendkívül személyre szabhatóak. A modellek kipróbálgatása után arra a következtetésre jutottam, hogy a Gemini 2.5 Pro Experimental 03-25 rendkívül jól teljesít a tesztfeladatra.

7.8. Egyéb említésre méltó modellek

7.9. Megjegyzés

A tényleges érzékenység mértékének megállapításához empirikus tesztekre van szükség.

7.10. Összegzés és következtetés

Egy kicsit mindegyik nyelvi modell úgy működik, mint egy hashelő algoritmus, abból a szempontból, hogy ha az inputot egy kis mértékben is változtatjuk, akkor az output teljesen más lehet. Ez akkor is igaz, ha a temperature 0-ra van állítva, magasabb temperature értékkel pedig főleg jellemző ez. Ez a viselkedés hasznos, ha a modellek fel tudjanak ismerni jelentős módosítószavakat, pl. tagadószócskákat, hangulatszavakat. A WiC feladatnál a mondatok fordított sorrendben való beadása ugyan nem változtat az input szemantikáján, ám a modellek sokszor tévesen más kontextusúnak gondolják azt.

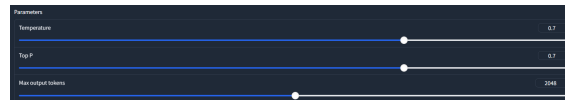
8. Kérdés-válasz szótár

A naiv kérdés-válasz készítési ötlet az lehetne, hogy a vizsgált szótő legyen a kulcs, és a kérdés a válasz. Ez azonban nem célszerű, sőt hibás, ugyanis több mondatpár is lehet az adathalmazban ugyanazzal a szótővel. Fontos, hogy a kérdés-válasz szótárban a kérdéseknek kell lenniük a kulcsoknak, a 'Yes' vagy 'No' válaszoknak pedig az értékeknek, és nem fordítva.

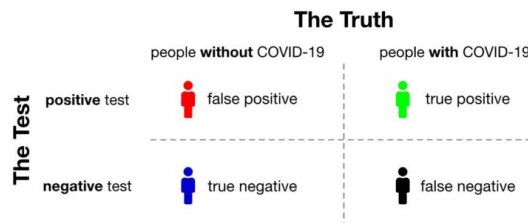
Egy sor a következőképpen néz ki: f'Does the word "word" mean the same thing in sentences "sentence_a" and "sentence_b"?': ['Yes' VAGY 'No']

8.0.1. 1.1.2.1 A beadott mondatok száma

Alapos tesztelés után azt találtam, hogy x példára oldják meg a nagy nyelvi modellek a legjobb pontossággal a feladatot.



12. ábra. Lmsys direct chat állítható paraméterei



13. ábra. Covid-19 példa a TP, TN, FP, FN alkalmazására

8.1. A nagy nyelvi modellek konfigurása a prompt beküldése előtt

9. Kiértékelési metrikák

9.1. Klasszifikációs módszer

A chatbotok teljesítményét egy tévesztési mátrix segítségével, klasszifikációs módszerrel értékeltem ki. Hogy ez mit is jelent a gyakorlatban, hozok egy példát. Covid esetén pozitív a teszt, ha a páciens beteg, negatív, ha egészséges. Továbbá igaz a predikció, ha helyesen ítéltük meg az egészségi állapotát, és hamis, ha helytelenül ítéltük meg. Ezeknek a kombinálásával kapjuk meg a TN, FP, TN, FN értékeket.

A WiC feladat esetében ezt úgy kell értelmeznünk, hogy pozitív a gold standard, ha a szó ugyanazt jelenti mind a két mondatban, negatív, ha eltérő dolgot. Igaz, ha helyesen ítélte meg a modell a szó jelentését és hamis, ha helytelenül ítélte meg.

Például:

- "True positive: Sick people correctly identified as sick"
- "False positive: Healthy people incorrectly identified as sick"
- "True negative: Healthy people correctly identified as healthy"
- "False negative: Sick people incorrectly identified as healthy"

9.2. Saját annotáció készítése

A WiC honlapján az áll, hogy az emberek körülbelül 80%-os pontossággal tudják a teszt adathalmazból vett, véletlenszerű példákról megállapítani, hogy a szó a 2 mondatban ugyanazt jelenti-e (a gold standard alapján). Ez nem túl magas, még úgy sem, ezek többsége ritka szó, és a mondatok nyelvezete is sokszor régies. A módszer az volt, hogy minden annotátor kapott 100 példát, és semmilyen segédeszközt nem használhattak.

9.3. Megvizsgált modellek

A Imarena.ai platformon az alábbi paraméterezéssel teszteltem a nagy nyelvi modelleket. a Temperature 0.0-ra állítva, a p-érték 0-re állítva, a max output tokens paramétert pedig 4096-ra állítva. Ez elengedhetetlen volt a megfelelő minőségű válasz eléréséhez. Részletek: Konzisztencia. A feladat megértéséhez elengedhetetlen, hogy pár az alábbi fogalmakkal tisztában legyek. Maximum Output Tokens és Context Window a Nagy Nyelvi Modellekben A "maximum output tokens" a modell által generált kimeneti szöveg leghosszabb engedélyezett hossza. Ez a paraméter kizárólag a válasz hosszát korlátozza, és nem befolyásolja a bemeneti adatok hosszát közvetlenül.

Fontos jellemzők:

A maximum output tokens értéke a kimeneti szöveg hosszának felső korlátját adja meg.

Ha ez az érték alacsony, a modell nem tud részletes vagy hosszú választ adni.

Ha túl magasra állítjuk, a modell felesleges vagy redundáns szöveget is generálhat. A "context window" az a maximális szöveghossz, amelyet a modell egyszerre képes figyelembe venni a bemeneti adatok feldolgozása során. Ezt általában tokenekben mérik, ahol egy token lehet egy teljes szó, szótöredék vagy akár egy írásjel is, attól függően, hogy a modell milyen tokenizálási technikát alkalmaz.

Fontos jellemzők:

A context window hossza határozza meg, hogy a modell milyen hosszú szövegeket képes értelmezni és kontextusba helyezni.

Ha a szöveg hossza meghaladja ezt a korlátot, a modell levágja vagy elhagyja a szöveg egy részét, ami információvesztéshez vezethet.

Például a GPT-3 esetében a context window alapértelmezetten 2048 token, míg a GPT-4 esetében ez 8192 vagy akár 32 768 token is lehet egyes verziókban.

A Imarena.ai webalkalmazáson kívül a nyelvi modelleket fejlesztő vállalatok hivatalos oldalain található modell legújabb verzióit is teszteltem, hogy mennyire jól oldják meg a WiC feladatot. Az alábbi táblázat tartalmazza a használt platformokat és a tesztelt

Large Language Models Table

1. táblázat. Hivatalos llm platformok és a rajtuk használt modellek

| | A | B | C | D |
|----|--------------------|--|--|---|
| | Model | Max input tokens | Max output tokens | |
| 1 | OpenAI GPT-4o | Each one has 4 "styles": Normal, Creative, Analytical and Formal | up to 250,000 tokens in the context window | |
| 2 | OpenAI GPT-4o mini | General | 4096 tokens per request | |
| 3 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 4 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 5 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 6 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 7 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 8 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 9 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 10 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 11 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 12 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 13 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 14 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 15 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 16 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 17 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 18 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 19 | OpenAI GPT-4o | General | 4096 tokens per request | |
| 20 | OpenAI GPT-4o | General | 4096 tokens per request | |

14. ábra. Nagy nyelvi modellek összehasonlítása

modelleket, platformonként.

Rengeteg interneten elérhető nagy nyelvi modellel lehet ingyenesen csevegni (szöveges promptot inputként beadni, melyre szöveggel tér vissza). Ezeknek a legfőbb előnye a lokális modell futtatással szemben, hogy nagyon gyorsak, és alig veszik igénybe az eszközünk erőforrásait - megjegyzendő, hogy egy gyenge minőségű általános célú nagy nyelvi modell futtatásához is ipari mennyiségű hardverre és erőforrásra van szükség.

Azonban sok hátrányuk is van. Sajnos azt kellett észrevennem, hogy ezek a személyes, vagy munkahelyi használatra tervezett platformok nem, vagy alig konfigurálhatóak. Emiatt alig javítható az általuk elért eredmény, és ha javul is, az inkább az emlékező képességüknek köszönhető. A modell másik felhasználó számára, illetve másik környezetben ugyanazokat a hibákat követi el, mint a személyes betanítás előtt. A másik probléma a nagy nyelvi modellek hivatalos platformjaival, hogy az algoritmus működésébe - vagy a mai modelleknél már úgy is mondhatjuk, hogy a nagy nyelvi modell *gondolkodásába* - nincsen megfelelő mértékű betekintése a fejlesztőnek. Ezt a problémát igyekeznek kiküszöbölni pl. a Gemini-flash-thinking verziók, vagy a DeepSeek-R1 "részletes magyarázatot nyújtó" modelljei, azonban ez sem ad kellő irányítást. Másik hátrányuk ezeknek az online chat API-knak, hogy folyamatosan adatot gyűjtenek a felhasználótól, és azt aktívan felhasználják a modelljeik tanítására, és akár harmadik félnek is kiadhatják. Emiatt semmilyen bizalmas adatot nem érdemes megadni egy ilyen nagy nyelvi modellel való chatben.

Megoldás Kvantált nyelvi modell letöltése és lokális futtatása. Megoldásként olyan nagy nyelvi modelleket kerestem, amelyek nyílt forráskódúak és elég kicsik ahhoz, hogy a személyi számítógépemen fussanak. Ennek a megtalálására a Hugging Face platformot használtam. A Hugging Face egy gépi tanulási (ML) és adatelemzési platform és közösség, amely segít a felhasználóknak gépi tanulási modellek építésében, telepítésében és betanításában. A Hugging Face-en számos ingyenes és open-source nyelvi modell található, amelyek elfutnak egy egyszerű asztali gépen. Ezek a kis méretük miatt könnyen félrevezethetők, de a célfeladat nem igényel általános tudást a modell részéről. Itt a hangsúly az angol szavak jelentésének a felismerésén van, erre a célra a kvantált modell tökéletesen megfelel. Hugging Face-re való bejelentkezés után elérhetővé válik 2 nagyon fontos

feature: - Bármelyik modellt kipróbálhatjuk a webes felületen keresztül, - Generálhatunk saját Access Tokeneket, amelyekkel letölthetjük a modelleket a saját eszközeinkre.

Az elérhető modelleket átnézve és a témavezetőmmel egyeztetve a gemma-2-2b-it modellt választottam. Ez a modell méret/minőség arányban egész jó, számos kvantált (a pontosság rovására kisebb méretűvé tett) variánsa létezik, és nagy fejlesztő közösség áll mögötte. Weben kipróbáltam a modellt. Beadtam neki a test‘adathalmazból készített teljes promptot (1400 kérdéssel).

Egy nagy nyelvi modell letöltése és telepítése közel sem egyszerű folyamat, még akkor sem, ha egy kvantált (a pontosság rovására kisebbé zsugorított) verzióját telepítjük. A Gemma-2-2b-it modell, és a hasonló, több millió paraméterből álló nyelvi modellek tanítása rendkívül drága és erőforrásigényes, és GPU nélkül nem igazán lehet egy átlagos személyi számítógépen elvégezni. A Huggingface-en szerencsére elérhetőek a gemma modellnek (és számos másik modellnek is) kvantált, azaz a minőség rovására kisebbé tett változatai.

Letöltés után nem érdemes a modellt csak CPU-n futtatni, mert nagyon lassú a tanítási folyamat és ugyanilyen lassan is válaszol. Előfordul, hogy fél óráig kell várni egy válaszra.

Tárhely követelmények

A gemma-2-2b-it modell a Hugging Face-ről a következő paranccsal klónozható le:

```
git clone https://<HUGGINGFACE\_USERNAME>:<TOKEN\_AZONOSITO>@huggingface.co/google/gemma-2-2b-it
```

Maga az előtanított, kvantált modell közel 5 GB tárhelyet foglal el a gépen. Emellett a használatához szükséges eszközök további kb. 10 GB tárhelyet igényelnek, ezt érdemes számításba venni megkezdés előtt.

Telepítési folyamat

1. Git LFS telepítése:

Git Large File Storage szükséges a 512 MB-nál nagyobb fájlok kezeléséhez

```
git lfs install
```

2. Python virtuális környezet létrehozása:

```
python -m venv gemma
```

3. Virtuális környezet aktiválása:

```
gemma\Scripts\activate.bat
```


git lfs install - Git Large File Storage downloader: 512 MB-nál nagyobb fájlok fel- és letöltéséhez Github és a lokális verzió között. Ezután a python virtuális környezetet kellett létrehozni az alkalmazáshoz: A 'python -m venv gemma' paranccsal. Ezután futtatni kellett az aktiváló szkriptet a 'gemma\Scripts\activate.bat' paranccsal. majd a 'python -m pip cache remove *' és a 'python -m pip cache purge' parancsokkal kellett biztosítani a pip cache-ének eltávolítását, majd megtisztítását. Ezután a setuptools telepítése jött a

```
pip install setuptools
```

paranccsal, majd a torch webes platformján a megfelelő operációs rendszer kiválasztása után, megfelelő bites, méretű és verziójú szoftverének az installálása következett a

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

paranccsal. Ezután a

```
pip install -U transformers
```

, amely sokkal egyszerűbbé teszi a gemma használatát Python kódban, majd végül a

```
pip install accelerate
```

segítségével az accelerate python könyvtár gyorsabbá teszi a modell GPU futtatásának a folyamatát.

A modell jellemzői

A lényege ugyanaz, mint mondjuk a chatGPT-nek. Inputként kap egy legfeljebb ny- 'hosszúságú szöveget, ahol az n' modellenként változik, majd erre fog válaszolni. Azért hívják modelleknek a modelleket, mert az emberi kommunikációt utánózzák (modellezik).

Előnyük, hogy nyílt forráskódúak, így tetszőlegesen módosíthatóak - habár alapos szakértelem kell hozzájuk. Emellett rendkívül biztonságosak: nincsenek rákötve a világhálóra, semmilyen szinten nem kommunikálnak a külvilággal, emiatt bármilyen privát adatot is nyugodtan be lehet nekik adni. Ezek nem általános célú modellek. Kvantáltak, ami azt jelenti, hogy mesterségesen le van csökkentve a méretük. Emiatt könnyen félrevezethetők és könnyen hallucinálnak is.

Nehézségek

A gemmának a hátránya, hogy nagy tárhelyet igényel és lassan válaszol, van, hogy 5 percre kell várni egy válaszra, azonban a célfeladatra megfelel. Általános célú feladatokra, például a házi feladat megoldására sokkal rosszabbul teljesít, mint egy GPT-4o, Claude 3.7 Sonnet, Grok vagy DeepSeek modelljei.

A gemma-2-2b-it nem általános célú modell, hanem specializált feladatokra optimalizált. Kvantált, ami azt jelenti, hogy mesterségesen le van csökkentve a mérete az alapjául szolgáló Gemma transzformerhez képest. Emiatt könnyen félrevezethetőek és könnyen hallucinálnak is. Nekem csak 1 célfeladatra kellettek, ezért alkalmasnak találtam.

10. Saját rendszer fejlesztése

10.1. Word2vec és BERT Elméleti háttér

Még mielőtt belekezdenék a tervezési gondolatmenetem kifejtésébe, hadd említsem meg, hogy miért a BERT-et választottam és preferáltam a Word2vec felett.

A BERT lényegében egy, a modellek reprezentációs képességét segítő szóbeágyazás, amely dinamikusan képes a mondatokhoz reprezentációt létrehozni. Ez nagy előny a régebbi, jól ismert statikus szóbeágyazásokkal szemben, mint például a Word2vec és a GloVe. Utóbbiak ugyanis egy szóhoz ugyanazt a szóvektort rendelik, bármilyen szövegkörnyezetben fordul is elő. A legnagyobb probléma ezzel, hogy sok szónak több, teljesen eltérő jelentése van, ám ezek a szóbeágyazások képtelenek modellezni szavak szemantikájának ezt a dinamikus természetét. A BERT ezzel szemben úgy hozza létre a mondatreprezentációt, hogy figyel a szövegkörnyezetre akár predikálás közben is. Ezzel lehetővé teszi, hogy ha egy szónak, vagy mondatnak lehet több értelme is, azt képes legyen a kontextus és a szó figyelembevételével megkülönböztetni.

10.2. Tervezés

10.2.1. Projekt tervezés fázis:

Fő szempontjaim voltak, hogy a projekt fájlstruktúrája könnyen értelmezhető legyen, kövesse a konvenciókat, könnyen lehessen benne tájékozódni, és logikusan legyen felépítve. Emiatt előre létrehoztam a szükséges mappákat, amelyekben a kódjaimat tárolni fogom. Az azonosítók elnevezését is igyekeztem rendkívül konzisztensen végezni: minden külső használatra szánt, úgynevezett 'utility' Python szkriptem a 'wic_' előtaggal kezdődött. Amelyik pedig belső használatra volt szánva, az nem kapott 'wic_' előtagot. Az ötletet többek között az Intel által fejlesztett OpenCV cv2 digitális képfeldolgozásra szolgáló függvénykönyvtárából merítettem, amely többek között Python nyelven is használható.

Itt ugyanis minden beimportálható függvény, enum, változó, konstans stb. a `cv2.` előtaggal kezdődik, amely rendkívül megkönnyíti a `cv2` azonosítóinak (Függvény-, Osztály-, Változó-, Konstans-, Metódus-, Paraméter-, Enum- és Névtérnevek) elkülönítését a saját azonosítók, és ezáltal elősegíti a kód olvashatóságát.

Ezenkívül a fájloknak rendszerint hosszú, akár egy teljes mondatnál felérő nevet adtam, hogy a fájl megnyitása nélkül is világos legyen, hogy mit csinál, miért felelős. Ezzel az elnevezési konvencióval lényegében a KISS -

Keep it simple, stupid

alapelve követtem, amely egy népszerű és bevált módszer arra, hogy sokszor primitívnek, vagy egyértelműnek tűnő dolgokat expliciten kiírunk, vagy primitív módon egymás után felsorolunk, ám ez az olvashatóságot jelentősen javítja. A hosszú azonosítónevek adása egyébként a Java és a legtöbb, objektum-orientált nyelvet használó programozók körében is elterjedt.

10.2.2. A projekt felépítése

A tervezés során számos lépést megtettem, hogy a projektem könnyen átlátható legyen:

- A harmadik féltől származó könyvtárak, csomagok és egyéb függőségek egységesen, *requirements.txt*-ben lettek definiálva. Ez egy Pythonos konvenció, amelyet én is követtem. A PyCharm eszközt kínál a projekt függőségeinek automatikus összegyűjtésére, és arra is, hogy a felhasználó egy gombnyomásra képes legyen ezeket egyszerre letölteni.
- **Modularitás:** A projektben a lehető legkevesebb fájlkon átívelő függőséget alkalmaztam, tehát a projekt kódja annyira moduláris, amennyire csak lehet. Ez különösen hasznos a projekt kódjának refaktorálásakor, illetve akkor, ha csak egy, vagy pár fájlt kellett használnom a projektből egy bizonyos feladatra, mert biztosította a program kis méretét és gyorsaságát.
- Kivétel erre a nem importálásra szánt fájljaim, ahol ténylegesen történik a függvények meghívása a saját függvénykönyvtár fájljaimból, és a konfigurációs beállítások (pl. Elérési utak) tárolására szolgáló fájljaim.

Mappák

- independent scripts mappa
- **llm_prompts mappa** Az `llm_prompts` mappában lévő szkriptekkel olyan promptokat hoztam létre, amelyekről jó eredményt lehet elvárni a nyelvi modelleken való promptolás esetén. Emiatt nyelvtanilag helyesek, a tokenek egymástól egyértelműen elkülönülnek, és szerkezetileg megfelelnek a nyelvi modellek elvárásainak:

a prompt elején van az utasítás, majd a kérdések következnek jól elkülönülve. A validáció és tesztelés érdekében készítettem pár nagyon egyszerű, és pár nagyon nehéz kérdést is a nagyon gyenge, illetve nagyon jól teljesítő modellek számára.

- **modules_and_data mappa** Ez a mappa a projekt futtatásához szükséges modulokat és az adathalmazokat tartalmazza. 2 részre oszlik:

Modulok: Ide tartoznak az adathalmaz tisztítását, felesleges, vagy használhatatlan adatok kiszűrését és egyéb adattranszformációkat végző Python szkriptjeim.

Data: Az adatok előfeldolgozása és átalakítása ebben a mappában történik, például tokenizálás, normalizálás vagy TF-IDF számítás.

- **solution mappa** Szintén 2 részre oszlik: Implementation: Ez a mappa tartalmazza a megoldásom végső implementációját és a projekt által generált eredményeket. Itt található a BERT-alapú megoldásom, valamint a generatív nyelvi modellek teljesítményének összehasonlítása.

Results: Az itt található fájlok a különböző modellek által generált válaszokat, tévesztési mátrixokat, statisztikákat és egyéb mérőszámokat tartalmazzák

- **use_modell_locally mappa** ipynb: GPU-s futtatáshoz használtam a lokális modellekre vonatkozó szkriptet. py: A webes API-on keresztül elérhető modelleknek pedig egyesével külön szkriptet hoztam létre, a kódot a fejlesztő cégek hivatalos oldalairól, illetve a Hugging Face transformers oldalairól szedtem le.

- **WiC_dataset mappa** A WiC (Word-in-Context) adathalmaz itt található

10.3. Implementáció

- BERT beágyazások
- Python algoritmusok, melyek Yes/No döntéseket hoznak
- Python algoritmusok, melyek előállítják a promptokat, amelyek inputként adhatók döntéshozásra nyelvi modelleknek

WiC_tf_idf_vectorizer_single szkript:

```
f'''Does the word \"word\" mean the same thing in sentences \"sentence1\" and \"sentence2\"?: 'answer', returns 2 values: - a single word 'Yes' or 'No' as answer, - a confidence score of Yes in %. 100% means it is a hundred percent true that they mean the same, 0% means they definitely mean different things.
```

Nagyon sokféle módszerrel kísérleteztem, a végső verzióban az alábbi 4 implementációt valósítottam meg.: - BERT, - NLTK (Natural Language Toolkit) Lesk, - Word2vec - Deep learning unsupervised reinforcement learning neural network

TODO: Kommentek A kommenteket egységesen angolul írtam mindenhol. Ez a legtöbb multinál amúgy is elvárás, gondoltam én sem teszek kivételt.

10.4. Tesztelés

A tervezett vizsgálatok során az alábbi nyelvi modellek kerülnek tesztelésre:

- **GPT-3.5, GPT-4o GPT-4o mini, gemini-flash, gemini-pro, gemini-flash-pro-thinking, Llama, Grok4, DeepSeek-r1, DeepSeek-r3, Claude Sonnet, Claude Haiku**
- **Gemma2-2b** és ennek kisebb méretű, kvantált változatai.
- **GPT-Neo**, a GPT-3 nyílt forráskódú változata.
- **OPT**, a Facebook által fejlesztett nagy nyelvi modell.
- **BLOOM**, a BigScience projekt többnyelvű LLM-je.
- Egyéb modellek, ha a kutatás során további releváns jelöltek merülnek fel.

10.4.1. Tesztelés Benchmarking módszerrel

Az összehasonlító teljesítményvizsgálatok az alábbi platformok segítségével történnek:

- LLM Arena **Large Language Model (LLM) API Playground by Retool**
- Google AI Studio
- Ollama asztali alkalmazás

Az eredmények kiértékelése során külön figyelmet fordítottam az egyes modellek által adott válaszok minőségére, a sorrendiségre való érzékenységükre, valamint az angol nyelvű érvelésük és döntéshozásuk pontosságára.

10.4.2. Konzisztencia

A kutatásom fő célja a modellek teljesítményének összehasonlítása a WiC benchmarkon volt, de emellett a konzisztenciájuknak megvizsgálását és összehasonlítását is kitűztem célul: Mennyire érzékenyek arra, hogyha a 2 mondatot felcseréljük? Tehát, ha a WiC adathalmazból előállított kérdésekből beadunk egyet a vizsgált modellnek, majd töröljük a modell memóriáját, és ezt a kérdést a 2 mondatot felcserélve is elküldjük a modellnek, akkor mekkora arányban egyezik a 2 válasz.

Ezt a kísérletet az alábbi módon végeztem el:

Először is, ahhoz, hogy egyértelmű eredményt kapjak a modell teljesítésére, először is minden véletlenszerűséget biztosító paramétert ki kellett kapcsolnom. Erre azért volt szükség, hogy ne a szerencsén múljon az, hogy a modell két válasza megegyezik, vagy eltér, hanem a modell felépítésén. A sztochasztikus faktorokkal ugyanis ugyanarra a kérdésre eltérő futtatáskor eltérő választ adhat egy modell, ami nem vezet hiteles a kutatáshoz.

Hogy ezt elérjem, azt az elterjedt módszert használtam, hogy a *temperature* értéket 0-ra állítottam, a *top-p* és *top-k* értékeket pedig 1-re. Ezzel elértem, hogy a sztochasztikus faktorokat nullára redukáljam.

10.4.3. Egy példa kérdés:

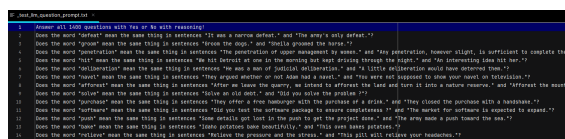
Egyenes sorrendben

Does the word "defeat" mean the same thing in sentences
"It was a narrow defeat." and "The army's only defeat."?

Fordított sorrendben

Does the word "defeat" mean the same thing in sentences
"The army's only defeat." and "It was a narrow defeat."?

A véletlen faktor kikapcsolása mellett az is fontos volt, hogy egyesével adjam be a kérdéseket. Amikor ugyanis a TODO-es és TODO-es ábrán látható formátumban, egyszerre N kérdéssel bombáztam a modelleket, akkor az is egy figyelembe veendő tényező volt, hogy az N kérdés milyen sorrendben lett feltéve. Ráadásul a modellek jellemzően csak 100 alatti kérdésre válaszoltak, a többit teljesen ignorálták. Ezzel szemben, ha minden kérdés esetén tiszta lappal indítottam, és új, független promptként adtam be, akkor ez a hatás nem tudott bezavarni. Emiatt tisztább a kísérlet, ha a modell nem egyszerre kapja meg mind az N mondatot, amiről a döntést várjuk, hanem N darab olyan kérdésként, amelyekben egyenként csak pontosan egy kérdés szerepel.



```

1. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
2. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
3. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
4. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
5. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
6. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
7. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
8. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
9. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
10. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
11. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
12. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
13. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
14. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
15. Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?

```

15. ábra. Sokmondatos prompt egyenes sorrendben beadott kérdésekkel

```
0 - Answer all 'n' questions with Yes or No with reasoning!
1 - Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
2 -
3 - Does the word "groom" mean the same thing in sentences "Groom the dogs." and "Sheila groomed the horse."?
4 -
5 - Does the word "defeat" mean the same thing in sentences "The penetration of enemy weapons is some." and "The penetration, however slight, is sufficient to complete the work."?
6 - Does the word "groom" mean the same thing in sentences "He will befall at one in his morning hat kept flying through the night." and "An interesting time hit her."?
7 - Does the word "defeat" mean the same thing in sentences "The war was a war of political determination." and "A little determination would have determined them."?
8 - Does the word "groom" mean the same thing in sentences "They argued whether or not I had a case." and "The army was supposed to show your news on television."?
9 - Does the word "defeat" mean the same thing in sentences "After he knew the party, he failed to prevent the land and turn it into a future reserve." and "The great the mountain."?
10 - Does the word "groom" mean the same thing in sentences "After an old dog." and "Did you solve the problem?"?
11 - Does the word "defeat" mean the same thing in sentences "They offer a few hundred" and "The purchase of a truck." and "They closed the purchase with a handshake."?
12 - Does the word "groom" mean the same thing in sentences "Did you visit the software package to ensure singleness?" and "The market for software is expected to expand."?
13 - Does the word "defeat" mean the same thing in sentences "Some details got lost in the rush to get the project done." and "The army had a good chance of the war."?
14 - Does the word "groom" mean the same thing in sentences "I have potatoes here beautifully." and "This was a good potato."?
15 - Does the word "defeat" mean the same thing in sentences "Before the end of the story." and "This will kill, please your responses."?
16 - Does the word "groom" mean the same thing in sentences "In the characteristic New York style." and "This style of dress is in demand."?
```

16. ábra. Sokmondatos prompt fordított sorrendben beadott kérdésekkel

```
Answer all 'n' questions with Yes or No!
Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
Does the word "groom" mean the same thing in sentences "Groom the dogs." and "Sheila groomed the horse."?
.
.
.
(n sor összesen)
```

17. ábra. Általános prompt formátum érvelés nélkül

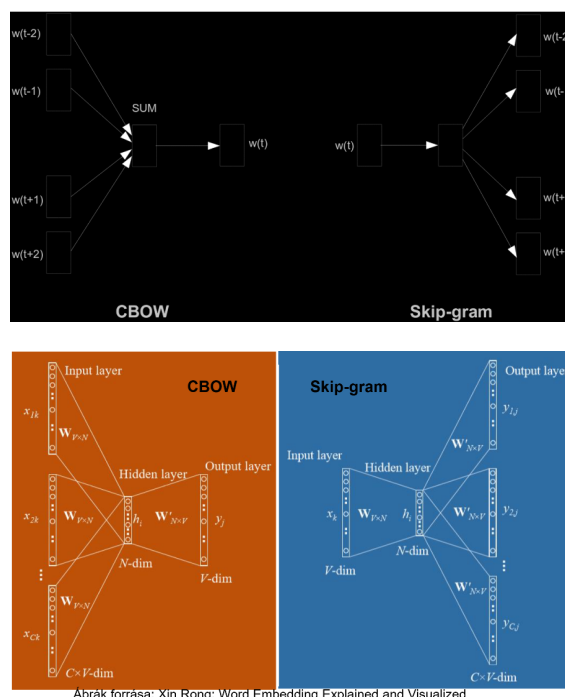
```
Answer all 'n' questions with Yes or No with reasoning!
Does the word "defeat" mean the same thing in sentences "It was a narrow defeat." and "The army's only defeat."?
Does the word "groom" mean the same thing in sentences "Groom the dogs." and "Sheila groomed the horse."?
.
.
.
(n sor összesen)
```

18. ábra. Általános prompt formátum érveléssel

A nagy nyelvi modelleket futtató szkriptem nyilvánosan megtekinthető, és a kísérlet elvégezhető Google Colabon, a mellékletben található linken.[TODO1].

Mindezek figyelembevételével elmondható, hogy a kiértékelésem módja:

- reprodukálható, azaz bárki meg tudja ismételni a kísérletet
- determinisztikus, tehát ugyanarra az inputra mindig ugyanaz lesz az output, nem függ a véletlentől.



2.1.1. word2vec

$$y(x) = softmax(W(W1x)) \quad (1)$$

- A word2vec célja, hogy a hasonló jelentésű input szavak hasonló outputot eredményezzenek. CBOW és Skipgram módszerek
- a és b szó jelentése minél hasonlóbb, $y(a)$ és $y(b)$ (eloszlás)vektorok annál inkább hasonlítani fognak
- CBOW: x „környező” szavak reprezentációi alapján akarjuk a „középső” szót előrejelezni $y(x)$ -szel
- Skipgram: x „középső” szó reprezentációja alapján akarjuk a „környező” szavakat előrejelezni $y(x)$ -szel

2.1.2. Vektortérmodell Szöveges tartalmak tömör reprezentációjára a vektortérmodell (VTM) nyújtja a legszélesebb körben használt megoldást. A modell minden egyes dokumentumot egy vektorral ír le, amelyben minden elem az egyes termek (általában szavak) előfordulását jelenti. Termek alatt a reprezentáció egységeit, alapesetben az írásjelek által határolt szavakat (unigram) értjük. Bizonyos módszerek több szóból álló kifejezéseket (n-gram) is alkalmaznak reprezentációként, amellyel általában jobban jellemezhetőek a dokumentumok, de jelentősen megnöveli a dokumentum feldolgozási idő- és tárigényét. A vektortérmodell egyik legelterjedtebb megvalósítása a bag-of-words (BoW) megközelítés, amely a dokumentumok reprezentálására kizárólag a termek előfordulási gyakoriságát veszi figyelembe, figyelmen kívül hagyva azok sorrendjét és kontextusát. Ennek egy

továbbfejlesztett változata a TF-IDF (Term Frequency - Inverse Document Frequency) súlyozás, amely nemcsak a termék abszolút előfordulását veszi figyelembe egy adott dokumentumban, hanem azok relatív fontosságát is az egész korpuszon belül. A TF-IDF segítségével a gyakori, de kevésbé informatív szavak (például névelők, kötőszavak) kisebb súlyt kapnak, míg a ritkábban előforduló, de tartalmilag releváns szavak nagyobb szerephez jutnak.

Az utóbbi évek fejlődésével az olyan mélytanulási modellek, mint a BERT (Bidirectional Encoder Representations from Transformers) és más transzformátor alapú nyelvi modellek még fejlettebb szövegreprezentációt kínálnak, figyelembe véve a kontextust és a szavak közötti összefüggéseket.

11. A 2 algoritmus eredményei és ábrák

11.1. Modell teljesítmény értékelésének módszere

A modell teljesítményét a pontosság (Accuracy), precizitás (Precision), fedés (Recall), F1-pontszám (F1-score) mutatókkal mértem, majd a *Matplotlib* és a *Seaborn* grafikai megjelenítő Python könyvtárak segítségével ábrázoltam a TP, FP, FN, TN értékek eloszlását.

A precizitás megadja, hogy hány százalékát teszik ki a TP és TN együtt az összes elemszámnak. A fedés azt ábrázolja, hogy a gold standard alapján igazak közül mekkora arányban tippelte azt a modell, hogy az tényleg igaz. Ezek képlete adja meg az F1-score-t, ami a precizitás és a fedés harmonikus közepe. Az F1 értéke a pontosság és a fedés harmonikus átlaga.

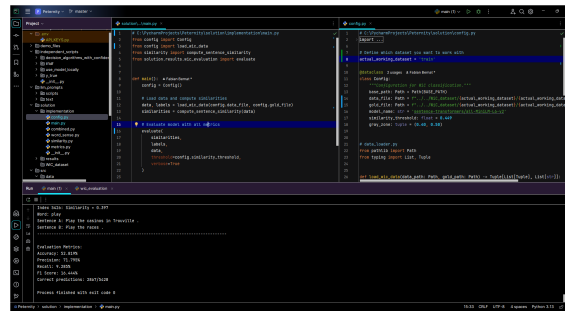
$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

19. ábra. Az F1 pontszám kiszámításának képlete

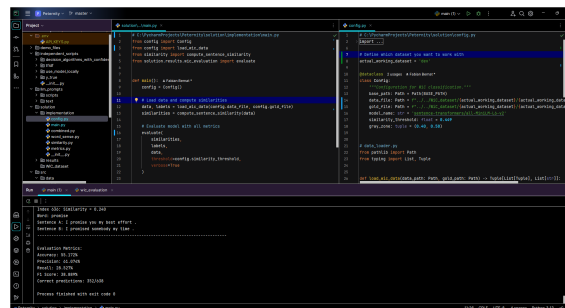
11.2. A részletes kimutatásokat tartalmazó megoldás

Pontosság, Precizitás, Fedés, F1-score és összesített statisztikák

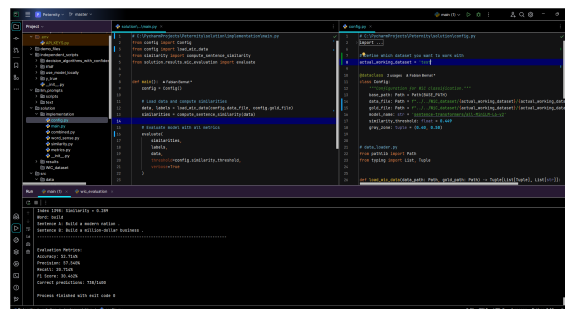
A *solution*, *implementation* és *results* mappában található standalone megoldás a *main.py* fájlban indítható. Ez a kiértékelő algoritmus a pontosság mellett a precizitást, a fedést és az F1 pontszámot is kimutatja.



20. ábra. Eredmények a train halmazra

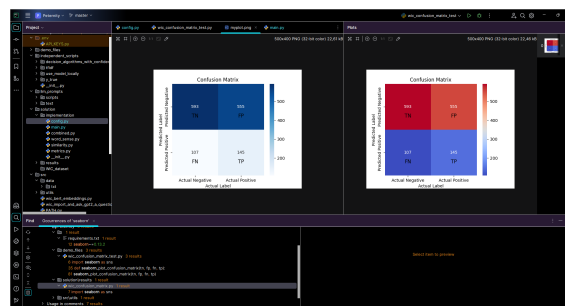


21. ábra. Eredmények a dev halmazra

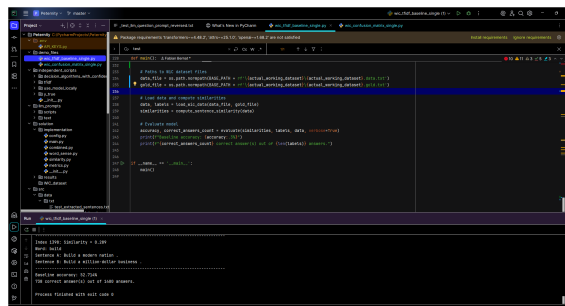
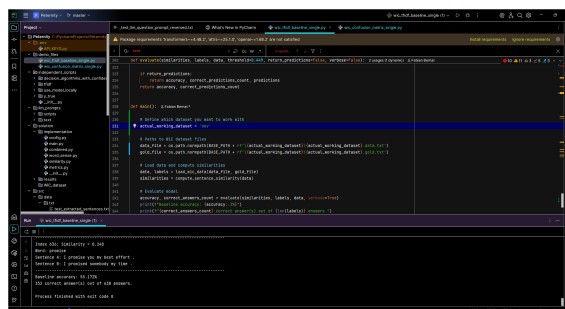
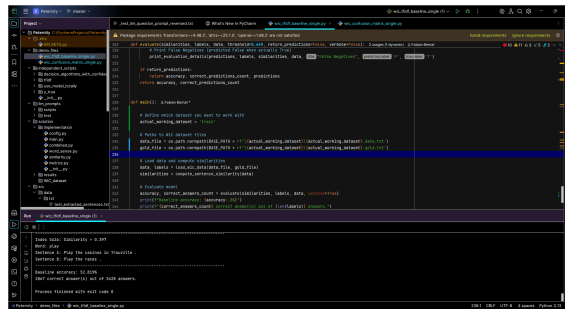


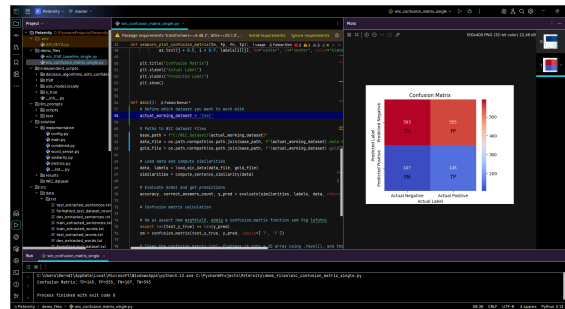
22. ábra. Eredmények a test halmazra

Ábrák



23. ábra. WiC tévesztési mátrix ábra





27. ábra. A kompakt megoldás tévesztési mátrixa a teszt halmazra.png

12. Az NLTK Használata

Az nltk egy természetes nyelvfeldolgozásra szánt Python könyvtár, amely számos eszközt biztosít. Rajta keresztül letölthető a WordNet lexikális adatbázis, a Wordnet szótár és a Lesk algoritmus.

A WordNet egy nagy lexikális adatbázis, amely angol nyelvű szavakat csoportosít úgynevezett "synset"-ekbe, amelyek kognitív szinonimákat, és szavak közötti szemantikai kapcsolatokat is tartalmaznak. Minden synset egy fogalmi jelentést reprezentál, és a különböző synset-ek különböző jelentéseket fejeznek ki. A Lesk algoritmus egy klasszikus WSD módszer, amely a vizsgált szó környezetében lévő szavak és a WordNet-ben szereplő definíciók közötti átfedés alapján határozza meg a legvalószínűbb jelentést.

A WordNet letöltését és elérhetőségének ellenőrzését a `download_wordnet_if_needed()` függvénnyel automatizáltam. A szinonimák és jelentések kinyerésére a `get_synonyms()` függvényt alkalmaztam, amely egy adott szóhoz tartozó összes lehetséges szinonimát visszaadja a WordNet segítségével. Az `expand_with_synonyms()` függvény pedig egy teljes mondat szemantikai kiterjesztését végzi el, ami segítségével a mondatok közötti szemantikai átfedések explicit módon vizsgálhatóvá váltak.

12.1. A WiC modell működése

A modell egy előtanított nyelvi modellt használ, amely képes a szóhasználatok közötti finom eltéréseket azonosítani. A bemenet két mondatból és a vizsgált szóból áll, míg a kimenet egy bináris döntés: azonos vagy eltérő jelentés.

12.2. Adatfeldolgozás és annotáció

Az adatok előkészítése során szükség van megfelelő annotációs technikák alkalmazására. Az adatkészleteket standard WiC benchmarkokból nyerjük, amelyek kézi annotációval lettek ellenőrizve.

12.3. Tanulságok, hipotézisek levonása

A WiC feladaton a természetes nyelvi szavak jelentésének komplexitásából adódóan szinte lehetetlen 100%-os pontosságot elérni, de ez nem is cél. A cél, hogy egy olyan WiC adathalmaz-kiértékelő szoftver készüljön, amely hosszú távon segít az embereknek, könnyen integrálható más projektekbe és széles skálán terjeszthető. A kvantált modellek, mint a Gemma-2-2b-it, kisebb méretük miatt könnyebben futtathatók korlátozott hardveres erőforrásokkal, de ez a méretcsökkenés a teljesítmény rovására megy (pl. lassabb válaszidő, pontatlanabb eredmények).

A Gemma modell bizonyos specifikus feladatokra (pl. Word-in-Context) megfelelően alkalmazható, de általános célú kérdések esetén jelentősen alulmúlhatja a nagyobb modelleket, mint pl. GPT-4o vagy Claude.

A modellek érzékenyek a bemeneti prompt formátumára, különösen a sorrendiséget tekintve. Ha egy modellt ugyanarra a kérdésre eltérő sorrendű mondatokkal kérdezzük meg, akkor a válaszokban jelentős eltérések lehetnek. Emiatt az egyenként feltett kérdések pontosabb eredményt adnak, mint a több kérdést egyben tartalmazó prompt.

A nyílt forráskódú és offline futtatható modellek előnye, hogy teljes adatvédelmi kontrollt biztosítanak, mivel nem kommunikálnak a külvilággal, így érzékeny adatok feldolgozására biztonságosabbak.

12.4. Tervek a jövőre nézve.

Legnagyobb vágyam ez a projekt után, hogy a XL-WiC problémát is mélyebben megismerjem, és lehetőleg magyar nyelvre is hasonló adathalmazokat találjak, és hasonló rendszert hozzak létre.

13. Összegzés

A WiC_decision_maker_model elérte a célját, hogy legalább 60%-os pontossággal el tudja dönteni a szavakról, hogy ugyanazt jelentik-e a 2 mondatban, vagy sem. Azt hiszem, sikerült átfogó elemzést nyújtanom a különböző LLM-ek WiC feladatokban nyújtott teljesítményéről. A projekt hosszú távon hozzájárulhat a szójelentés-felismerő rendszerek fejlesztéséhez és finomhangolásához. A dolgozatomat úgy állítottam össze, hogy nyomon követhető és reprodukálható legyen, ezért nyílt forráskódú eszközökre (pl. Hugging Face, NLTK) és nyilvános adathalmazokra támaszkodtam.

14. Nyilatkozat

Alulírott Fábián Bernát, 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 TVSZ 4. sz. mellékletében leírtak szerint kezelik.

Szeged, 2025. május 5.

.....

aláírás

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani szüleimnek, barátaimnak, tanítóimnak, tanárainak, egyetemi tanárainak, akik végigkísértek utamon tanulmányaim alatt, és hozzásegítettek végső soron, hogy sikeres szakdolgozatot készíthessek. Köszönetet szeretnék továbbá nyilvánítani szüleimnek, testvéreimnek és barátaimnak, hogy mindenben támogattak.

Végül, de nem utolsó sorban köszönetet szeretnék mondani **témavezetőmnek, Berend Gábornak**, hogy konzulensként és témavezetőként segített a szakdolgozatom megírásában.

15. Mellékletek

16. Hivatkozások és források

Az alkalmazás fejlesztése és elemzése során az alábbi forrásokra támaszkodtam:

- WiC adatbázis
 - Gemma modell konzisztencia kiértékelő szkript
 - Gemma2-2b-it modell
 - Kvantált modellek
 - Kvantálás elméleti háttér
 - A nagy nyelvi modellek definíciója:
Nagy nyelvi modell (LLM: Large Language Model) - YouTube
2. https://hu.wikipedia.org/wiki/Nagy_nyelvi_modellNagy nyelvi modell - Wikipédia
 3. Mik a nagy nyelvi modellek (LLM-ek)? - Microsoft Azure
 4. Mi az a nagy nyelvi modell? - Lexiq
 5. Nagy Nyelvi Modellek (LLM) - Mi ez és hogyan alakítja a jövőt?
 6. [PDF] A ChatGPT és más nagy nyelvi modellek (LLM-ek) alkalmazásának ...

Hivatkozások

[1] Irodalomjegyzék

Feladatleírás, valamint forrás a tanító, fejlesztő- és tesztelő adatbázishoz: WiC: The Word-in-Context Dataset Mohammad Taher Pilehvar honlapja: About Mohammad Taher Pilehvar WiC tematikájú CodaLab versenyek Online LLM referenciák: Chatbot Arena (formerly LMSYS): Free AI Chat to Compare Test Best AI Chatbots Claude 3.7 Sonnet és Haiku Anthropic hivatalos dokumentáció Gemini ingyenes chat API DeepSeek-R1 ingyenes chat API Google AI Studio

Lokális nyelvi modell futtatáshoz használt toolok: Windows natív C/C++ compiler CUDA Toolkit 12.8 Update 1 Downloads | NVIDIA Developer google/gemma-2-2b-it · Hugging Face PyTorch

- [2] @misc{wikipedialeskalgorithm, author = Wikipedia contributors, title = Lesk algorithm — Wikipedia, The Free Encyclopedia, year = 2025, url = https://en.wikipedia.org/wiki/Lesk_algorithm, @inproceedings{pilehvar-camacho-collados-2019-wic, title = Nagynyelvmodell-eketsszehasonltatform@misc{chiang2024chatbot, title = Chatbot Arena