Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Sáfrán Gergely

SZÖVEGES TARTALOM CÉLZOTT VÉLEMÉNYANALÍZISÉNEK JAVÍTÁSA NAGY NYELVI MODELLEK FINOMHANGOLÁSÁVAL

TDK Dolgozat

KONZULENS

Dr. Papp Dávid

BUDAPEST, 2023

Tartalom

Összefoglaló

Dolgozatomban a többcélpontú célzott véleményanalízis témakörben végezett kutatásomat és kísérleti eredményeimet mutatom be. Véleményanalízist széles körben használnak, üzleti és kutatási céllal, természetes nyelvfeldolgozás segítségével. A témában többek között arra keresnek választ, hogy egy adott szöveg hangvétele milyen, vagy a célzott véleményanalízis esetén, a hangvétel egy kiválasztott alany felé milyen jellegű. Különböző kategóriákat lehet megállapítani, például lehet pozitív, negatív, neutrális érzelmeket kifejezni, de összetett érzelmek vizsgálata is lehetséges. Célom az volt, hogy megvizsgáljam, nagy nyelvi modellek segítségével milyen minőségű véleményanalízis végezhető.

Nagy nyelvi modellnek olyan neuronhálót nevezünk, amely komplex architektúrája milliós, billiós nagyságrendű paramétert tartalmaz, ilyen például a ma legtöbbet használt transzformer architektúra. Az előtanításhoz TB nagyságrendű szöveges állomány felhasználása szükséges, és önfelügyelt, vagy félig felügyelt módon végezhető. Az így felépülő modellek különböző sztenderd nyelvi feladatok során minden korábbi módszernél jobb eredményeket érnek el.

Kutatásom során három eltérő megközelítés szerint végeztem többcélpontú célzott véleményanalízist. Az analízist a NewsMTSC adathalmazon, a GPT-3 modellcsalád több változatának felhasználásával végeztem, amelyek hatékony működése érdekében az úgynevezett *prompt engineering* (“parancs tervezés”) technikát alkalmaztam. Először megvizsgáltam, hogy a zero-shot learning milyen pontosságot ér el a különböző modellekkel, majd a következő lépésben bemenetként példa mondatokat adtam, few-shot learning módszerrel próbálkoztam. Végül a modellcsalád több elemét példa prompt – példa válasz párokból épített tanító halmaz felhasználásával finomhangoltam (fine-tuning).

Az eredmények azt mutatták, hogy megfelelő prompt alkalmazása esetén a Curie modell finomhangolt változata képes a korábbi state-of-the-art többcélpontú célzott véleményanalízis eredményeinek felülmúlására. Ez a NewsMTSC teszt halmazain rendre 86,47% és 87,33% osztályozási pontosságot jelentett, míg a szerzők 2021-ben publikált legjobb mérései 83,8% és 84,6% voltak, amelyeket a RoBERTa-GRU modell használatával sikerült elérniük.

Abstract

In my thesis, I present my research and experimental results on the topic of multi-target targeted sentiment analysis. Senitment analysis is widely used for business and research purposes using natural language processing. Among other things, it is used to find out the tone of a text or, in the case of targeted sentiment analysis, the nature of the tone towards a selected subject. Different categories can be identified, for example, positive, negative, and neutral emotions can be expressed, but complex emotions can also be examined. My aim was to investigate the quality of sentiment analysis using large language models.

We call a large language model a neural network whose complex architecture contains millions or trillions of parameters, such as the transformer architecture most used today. The pre-training requires the use of text files of size TB and can be performed in a self-supervised or semi-supervised manner. The resulting models perform better than any previous method on a variety of standard language tasks.

In my research, I conducted a multi-target targeted sentiment analysis using three different approaches. The analysis was conducted on the NewsMTSC dataset using several versions of the GPT-3 model family, on which I used prompt engineering to ensure their efficient operation. First, I investigated the accuracy of zero-shot learning with different models, and next, I input example sentences, using few-shot learning. Finally, I fine-tuned several elements of the model family using a training set constructed from example prompt - example response pairs.

The results showed that, with appropriate prompts, a fine-tuned version of the Curie model can outperform previous state-of-the-art multiobjective targeted sentiment analysis methods. This resulted in classification accuracies of 86.47% and 87.33% on the NewsMTSC test sets, while the authors' best published measurements in 2021 were 83.8% and 84.6%, respectively, achieved using the RoBERTa-GRU model.

1 Bevezetés

Véleményanalízist gyakran végeznek a természetes nyelvfeldolgozásban, pl. tőzsdei hírek elemzése LLM-ek (large language model) segítségével [1], felhasználói visszajelzések (review-k) elemzésére, közösségi média bejegyzések feldolgozására, pl. twitter bejegyzések, tweetek elemzésére [2]. Ennek egyik hajtóereje a cégeknél keletkező nagy mennyiségű szöveges adat feldolgozása, üzleti szempontból, illetve egy másik, a kutatási céllal való vizsgálat.

Az egyik lehetséges üzleti szempont a különböző márkák megítélésének [3], hírnevének monitorozása a termékek és szolgáltatások fogyasztói visszajelzéseinek elemzésével. Egy másik felhasználása a véleményanalízisnek hasznos lehet, a befektetők számára, a piaci megítélés felderítésére és a megfelelő válaszlépés kidolgozására. Ezek mellett a politikában is alkalmazható, például a kampányidőszakban a jelöltek népszerűségének vizsgálatára.

Többféle szempont szerint lehet vizsgálni az adott szöveget, egyes esetekben a feldolgozás során csak arra vagyunk kíváncsiak, hogy egy mondat, vagy szövegrészlet milyen hangvételű, pl. pozitív, vagy negatív, de lehet célzott analízist is végezni, ahol egy, a szövegben szereplő alany felé irányuló viszonyt vizsgálunk meg. Bemutatok egy példát a véleményanalízisre:

> *‘Hillary Clinton blamed the Democratic National Committee, Facebook, and conspiracy site Infowars Wednesday for her election defeat during an interview in which she pointed at a total of 18 alleged/guilty parties for her big loss.'*
>
> *Alany 1: former Secretary of State Hillary Clinton*
>
> *Negatív az alany irányában.*

E mellett lehet többalanyú vizsgálatot is végezni, akkor a fenti példa a következőképpen módosul:

> *‘Hillary Clinton blamed the Democratic National Committee, Facebook, and conspiracy site Infowars Wednesday for her election defeat during an interview in which she pointed at a total of 18 alleged/guilty parties for her big loss.'*

Alany 1: former Secretary of State Hillary Clinton

Alany2: Facebook

Negatív alany1 irányában.

Negatív alany2 irányában

A bevezetés után a második fejezetben megvizsgálom, hogy hogyan lehet neurális hálózatokkal szekvenciális adatot, például szöveges adatot feldolgozni, kezdve a rekurrens hálózatokkal, majd a transzformer hálózatokat bemutatva. Kitérek a nagy nyelvi modellek tanítására, és az elérhető pontosságra a véleményanalízis terén.

A harmadik fejezetben leírom a vizsgált adathalmazon az elvégzett elemzést, kezdve az adathalmaz bemutatásával, a korábbi kutatás eredményeivel, majd a prompt engineering magyarázatával. Végül pedig kiértékelem a különböző módszerekkel kapott eredményeket, és levonom a következtetéseket.

A negyedik fejezetben röviden összefoglalom az eredményeket.

2 Véleményanalízis neurális hálózatokkal

2.1 Rekurrens hálózatok

A hagyományos neurális hálózatok[1] széles körben alkalmazhatók különféle problémák, illetve részfeladatok megoldására. Hátrányuk azonban, hogy egy korábban beérkező soros jellegű információt, például egy mondat eddig elolvasott részét nem képesek eltárolni, és továbbadni, majd kötni az újonnan beérkező mondatrészhez.

Az RNN (Recurrent neural network) hálózatok a hagyományos neurális hálózatokhoz hasonlóan épülnek fel, azonban a hálózatba egy új funkció is kerül, a hálózat kimenetét visszacsatoljuk (a rejtett állapot előző kimenetét) az aktuális rejtett állapotához a következő lépésben. Az így felépülő hálózat alkalmas lesz szekvenciális adatsorok modellezésére. A neurális hálózat minden egyes időpontban adott bemenetre kimenetet ad vissza, majd ezt a kimenetet a t+1. időpontban megkapja bemenetként.

Ez a működés alapvetően szekvenciális jellegű [4], így kevésbé lehet párhuzamosítani, de különböző módszerekkel azonban lehet javítani a hatékonyságát. Probléma az ilyen adatsorok soros feldolgozásánál, hogyha az adatok közötti összefüggéseket egymástól távol lévő adatrészek között kell vizsgálni, a memóriaigény nő. Ez jellemző az RNN-ekre és a LSTM hálózatokra.

Más természetes nyelvfeldolgozási feladatokhoz hasonlóan a TSC is jelentős fejlődésen ment keresztül az utóbbi években. Ez a jelentős fejlődés a korszerűbb nyelvi modellek (2019), azon belül is a transzformer architektúrájú mélytanulásos modellek létrejöttéhez köthető [4] (2017). Ezek a transzformer hálózatok a korábban említett memóriaproblémára is hatékonyabb megoldást nyújtanak.

2.2 Transzformer architektúra áttekintése

A manapság használt legnépszerűbb nagy nyelvi modellek, BERT, GPT, LLAMA, PaLM mind a transzformer architektúrára épülnek. A BERT és a GPT 2.0 modellek is nagy potenciált mutattak, de a köztudatba a ChatGPT révén kerültek be a transzformerek. Nyelvi, fordítási, szövegértelmezési,

[1] https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b

feladatokon és teljesítményértékeléseken is kiváló eredményeket érnek el. Példának említem, a SAT teszten elért eredményeket, illetve a GPT-4 sikeres BAR vizsgáját. Az OpenAI belső tesztelése során a GPT-4, a SAT teszten 1410 pontot ért el, amely az emberi teljesítménnyel összehasonlítva a tesztet végzők 94%-ánál jobb. Az LSAT teszten ugyanez a mutató 88% volt, míg a BAR vizsga esetében 298 pontot ért el, ami 90%-os értéknek felel meg.

Ezek mellett a GPT-4 multimodális modell, vagyis többféle adattal is képes dolgozni, képekkel és szöveggel egyaránt. Amennyiben a modell erre utasítást kap, más interfészekkel is kapcsolatba lép[2], így aktuális információkhoz juthat, matematikai műveleteket végezhet, és más harmadik féltől származó szolgáltatásokkal is közreműködhet. Több alkalmazás is épül nagy nyelvi modellekre, a GPT-t integrálták a Microsoft Office termékeibe, a Bing keresőbe, a Github copilot alkalmazásba, és a most megjelenő Microsoft Copilotba.

2.2.1 Párhuzamosítás

Az adatfeldolgozás hatékonyabbá tétele érdekében főként a szekvenciális jellegű működést kell felváltani párhuzamos feldolgozással, ahol a bemenetek és kimenetek összes pozíciójára egyszerre történnek meg a számítások. Erre a célra jöttek létre a ByteNet, ConvS2S hálózatok, amelyek hagyományos neurális hálózatokat használnak építőelemként. A ByteNet olyan egydimenziós konvolúciós hálózat, mely képes a bemenet méretével lineáris időben nyelvfordítást végezni, és ehhez kevesebb memóriát használ fel. A különböző súlyok, pozíciós kódok, kulcsok, lekérések, és értékek feldolgozása összeadások és mátrixszorzások végzésével, párhuzamosan, GPU-n gyorsan elvégezhetően történik. A transzformer modellek legfőbb előnye is a párhuzamosíthatóság javítása, a különböző szekvenciák és ezekből képzett vektorok egyszerre történő feldolgozásával.

2.2.2 Transzformer modellek mérete

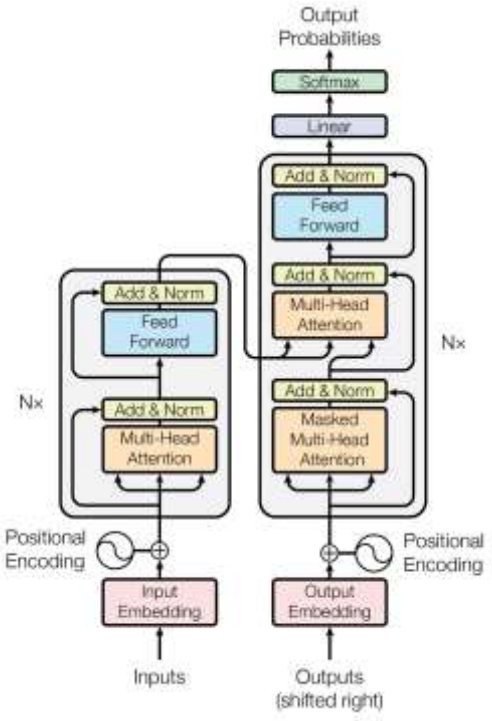
A nagy nyelvi modellek jelentősen különböznek a modellek méretében, amit egyfelől jellemez a paramétereik száma. A legtöbb modellnek különböző méretű változatai léteznek, pl. a GPT-3 esetében a legkisebb modell 125 millió paraméterrel rendelkezik, míg a legnagyobb 175 milliárd paraméterrel. A ma elérhető legnagyobb publikus modell a BLOOM, melynek a GPT-3 modellhez

[2] https://openai.com/blog/chatgpt-plugins

hasonlóan 175 milliárd paramétere van. A legnagyobb nem publikus modellek, mint például a GPT-4, vagy a PanGu paramétereinek száma meghaladja az egybilliót.

Általában lebegőpontos számokat használnak a paraméterek leírásához, float32, vagy float16 típust, így egy paraméter mérete 4 vagy 2 byte lesz. Ahhoz, hogy ezeket a paramétereket, a gyors elérés érdekében, egyszerre a memóriába pl. GPU memóriába elhelyezzük, egy 175milliárd paraméteres davinci-03 modell esetében, 2 byteos paraméterméretttel számolva kb. 350GB VRAM szükséges. Ekkora mennyiségű videómemória általában nem áll rendelkezésre az otthoni felhasználók számára, de ipari és céges környezetben, illetve a felhőben jellemzően Nvidia H100-as vagy A100-as GPU-kon elhelyezhető egy ilyen nagy modell. Mind a H100-as mind az A100-as típusú videókártyának létezik 80GB memóriával gyártott változata, ezekből 5db elegendő méretű memóriával rendelkezik, vagy jellemzően egy 8 GPU-s clusteren is futtatható a modell.

2.2.3 Architektúra



1. Ábra: A transzformer architektúra felépítése [4]

Az egyes ábrán a transzformer architektúra [4] látható, amit két fő részre lehet bontani, egy kódoló (az ábrán a bal oldali struktúra) és egy dekódoló részre (jobb oldal). A kódoló hat egyforma rétegből

áll, ahol minden réteg két alrétegre válik szét. Az első alréteg egy multi-head (több fejű) self-attention (ön-figyelő) mechanizmust valósít meg. Ez a multi-head rendszer „h" különböző lineáris vetítését kapja meg a kulcsoknak, lekérdezéseknek és értékeknek (keys, queries, values), amelyekből aztán a fejek párhuzamosan kimeneteket állítanak elő, majd egy végeredményt.

A második alréteg egy FFN (feed-forward network), vagyis egy teljesen összekapcsolt előrecsatoló hálózat, amely két lineáris transzformációt valósít meg, köztük ReLU aktivációs függvénnyel.

$$FFN(x) = ReLU(\mathrm{W}1x + b1)W2 + b2 \quad (1)$$

Az 1-es egyenlet fejezi ki a FFN hálózat matematikai leírását, W-vel jelölve a súlyokat, b-vel a bias-t, és a ReLU aktiváció is szerepel benne.

A hat réteg mindegyike ugyanazt a lineáris transzformációt alkalmazza a bemenet szavaira, de mindegyik réteg különböző W (weight) és b (bias) értékeket használ. Emellett minden alréteget egy normalizációs réteg követ, ami normalizálja az alréteg bemenete, és az alréteg által generált kimenet összegét. Fontos megjegyezni, hogy a transzformer architektúra alapesetben nem képes a szavak sorrendjét feldolgozni. Ezt az információt külön juttatják be a modellbe, a szavak pozíciós kódolásával. A pozíciót kódoló vektorok ugyanolyan dimenziószámúak, mint a bemeneti beágyazások (a tokenekből készült vektorok) és különböző frekvenciájú szinusz és koszinusz függvényekből készülnek. Az így keletkező vektorokat egyszerűen hozzáadják a bemeneti beágyazásokhoz, így a modell képes lesz ezt az információt is figyelembe venni.

2.2.4 Dekódoló

A dekódoló felépítése hasonlít a kódolóéhoz, ugyanúgy hat egyforma rétegből áll. Ezek a rétegek azonban, kettő helyett, három különböző alrétegre oszlanak.

Az első alréteg bemenetként megkapja a dekóder kupac előző kimenetét és egy több-fejes önfigyelmi mechanizmust valósít meg rajta. A kódolót olyannak tervezték, hogy a bemenet minden szavát figyelembe vegye, azonban a dekódoló úgy van módosítva, hogy csak a megelőző szavakra figyeljen. Ezért az i. pozícióban levő szó előállítása kizárólag a sorrendben megelőző szavak kimeneteitől függ. Ezt a párhuzamos feldolgozású, többfejű mechanizmusnál, maszkokkal valósítják

meg. A maszk a Q (query) és K (key) mátrix szorzatából képzett új mátrixban a főátló fölötti értékeket -∞-re állítja, ennek a maszknak a működését a 2-es egyenlet mutatja be a mátrix elemein.

$$mask(QK^T) = mask\left(\begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{m1} & e_{m2} & \cdots & e_{mn} \end{bmatrix}\right) = \begin{bmatrix} e_{11} & -\infty & \cdots & -\infty \\ e_{21} & e_{22} & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ e_{m1} & e_{m2} & \cdots & e_{mn} \end{bmatrix} \quad (2)$$

A második réteg a kódoló első alrétegéhez hasonló többfejű önfigyelmi mechanizmust valósít meg. A dekóder bemenetként megkapja a „q" lekéréseket, az előző alrétegből, a „k" kulcsokat és „v" értékeket pedig a kódoló kimenetéből. Így a dekódoló a bemeneti szavak mindegyikét figyelembe veszi.

A harmadik alréteg egy teljesen összekötött előre csatoló hálózatot valósít meg (FFN), hasonlóan a kódolóéhoz. A három alréteg körül is vannak maradék kapcsolatok és a rétegeket normalizációs réteg követi. Hasonlóan a kódolóhoz, a dekódoló bemenetéhez is hozzáadják a pozíciós információkat.

2.2.5 Tanítás

A nagy nyelvi modellek tanítása időigényes és költséges feladat. A 175 milliárd paraméteres GPT-3 modell tanítása 1db [3] Tesla V100 GPU-n, közelítőleg 355 évet venne igénybe és 4,6 millió dollárba kerülne, (az aktuális ár változhat).

Emellett rendkívül nagy szöveges adatbázisok szükségesek hozzá, a GPT-2 1,5 milliárd paraméteres modellhez 10 milliárd tokent, kb. 40GB, internetről gyűjtött, szöveges adatot használtak fel. A GPT-3 modellek tanításához [5] felhasznált adatok pontos összetétele és mennyisége nem áll rendelkezésre, egyes források szerint 292 milliárd tokent használtak, más forrás szerint a tokenek száma 499 milliárd, melyben a webről automatikus kigyűjtött szövegek, több könyveket tartalmazó adatbázis, és a wikipédia teljes egésze megtalálható. A tanító adatok mennyiségének a modellek méretéhez kell igazodnia, vagyis nagyobb paraméterszámú modellek tanításához több token szükséges.

[3] https://lambdalabs.com/blog/demystifying-gpt-3

Adathalmaz	Tokenek Száma (milliárd)
Common Crawl	410
WebText2	19
Books1	12
Books2	55
Wikipedia	3

1. Táblázat: GPT-3 modell szöveges tanítóhalmaza

Ebben a táblázatban a Common Crawl halmaz 410 milliárd tokenszámmal szerepel, ami 570GB adatnak felel meg. Ez a mennyiség a teljes Common Crawl (kb. 45TB tömörített szöveg) tanító halmaznak a szűrt változata.

Össze lehet vetni a különböző modellek esetében a tömörítési rátájukat, vagyis, hogy a paraméterszámuk hogyan aránylik a tanító tokenek számához. Ez a GPT-2 1.5B esetében 10/1,5 vagyis 6,66-os tömörítési ráta, a GPT-3 175B esetében kb. 500/175 vagyis 2,85. Ez felveti annak a kérdését, hogy mivel a GPT-3 tömörítési aránya kisebb, a modell vajon jobban memorizálja-e a tanító adathalmazt, és innen ered a jobb teljesítménye.

2.3 In-context learning

2.3.1 Zero-Shot learning

A modell számára zero-shot esetben [5] kizárólag egy szöveges leírást adunk a feladatról, példamegoldást nem. Ez a módszer egyszerű, azonban a legnagyobb kihívást is ez adja a modell számára. Ilyenkor a modell nem képes kihasználni a hamis korrelációkat az adatban, mivel nincs elegendő bemenő információ ehhez. A zero-shot feladatleírás sokszor az emberek számára is nehéz, ugyanis nincs megadva az elvárt formátum, nem derül ki pontosan melyik adatokra van szükség és melyek elhanyagolhatók. Mindezek mellett sokszor ez a leírás hasonlít legjobban ahhoz, ahogy mi emberek feladatokat megoldunk.

2.3.2 One-shot learning

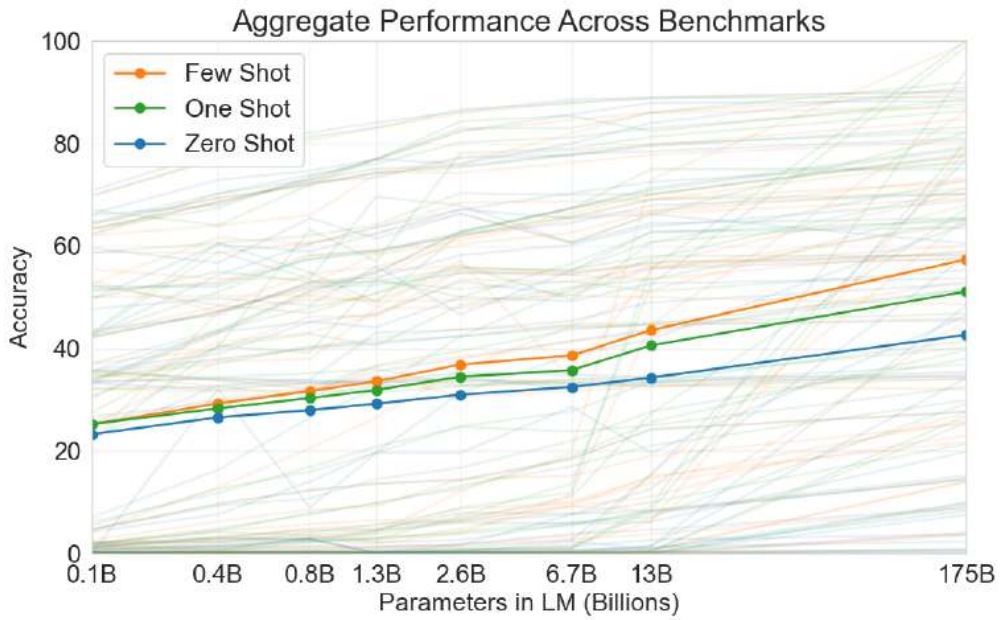
Hasonlóan a few-shot módszerhez itt is példákat adunk meg a modellnek, a feladat szöveges leírása mellé, ahhoz, hogy az általunk elvárt kimenetet adja vissza. A fő különbség, hogy itt egyetlen példa megadása történik, azonban ezt megkülönböztetjük a zero-shot learning-től, ahol nem adunk példamegoldást.

2.3.3 Few-shot

Few-shot learning alatt olyan eljárást értünk, melynek során a modell számára példákat mutatunk, jellemzően (2-10) darab nagyságrendben, és a modell súlyai nem kerülnek frissítésre. A példák emellett bele kell férjenek a modell által maximálisan feldolgozható bemeneti méretbe. Ilyen tanulási feladat lehet például a nyelvi fordítás, ahol megadjuk példa párokként mindkét nyelven a fordítandó szót, tehát az elvárt bemenet és kimenet párokat. A fő előnye ennek a módszernek, hogy jóval kevesebb címkézett adat szükséges, az adott feladat specifikálásához, mint a finomhangolás esetében, de a modellek teljesítménye általában javul. Azonban a modellek few-shot teljesítménye messze elmarad a legkorszerűbb finomhangolt modellekétől, a teljesítményértékeléseken.

2.4 Fine-tuning

Fine-tuning eljárást gyakran használnak egy előtanított modell esetén. Ennek az az oka, hogy például a természetes nyelvi feladatokon nagyon jó teljesítményt nyújtanak a finomhangolt modellek. A módszer során az előtanított modellt egy a feladatnak megfelelő, felügyelt adathalmazon tovább tanítjuk, így a súlyai frissítésre kerülnek. Ehhez jellemzően több ezer, vagy több tízezer címkézett tanító példa szükséges. Legfőbb hátránya a módszernek, hogy minden egyes feladathoz szükség van egy új nagy címkézett adathalmazra, amit nehéz és költséges előállítani. Hátrány továbbá, hogy ha más feladatot adunk a modellnek, akkor nehezen általánosít. További hátrány, hogy a modell a tanító adatban rejtetten megtalálható furcsa összefüggéseket, korrelációkat is kihasználhat, így torzítva a teljesítményértékelésen mutatott eredményt.



2. Ábra: Modellek aggregált teljesítménye a few, one és zero-shot eljárások szerint [5]

A 2-es ábrán egy olyan kutatás eredményeinek összesítését figyelhetjük meg, ahol a különböző méretű modellek teljesítményértékelései szerepelnek, egészen a legkisebb, 100 millió paraméteres modelltől a legnagyobb, 175 milliárd paraméteresig. Ezek az értékelések 42 különböző feladaton lettek elvégezve, majd egy grafikonon összefoglalták őket. Háromféle módszerrel vizsgálták a modelleket, ezek a Few-Shot, One-Shot, és Zero-Shot. Látható, hogy a zero-shot eljárás pontossága a modell méretével folyamatosan nő, míg a few-shot eljárás ennél is gyorsabban, ami arra utal, hogy a nagyobb modellek hatékonyabbak az in-context learning-ben.

2.5 Vélemény analízis pontosságának vizsgálata

A véleményanalízis vizsgálati módszereknek széles skálája van, melyek az informatikai tudományok több területéről származnak. Egyes eljárások gépi tanulást használnak, amihez pl. felügyelt tanulással, címkézett adatokat használnak fel. Lehet más megközelítéssel, pl. szótárakkal dolgozni, melyekben minden előre meghatározott szóhoz valamilyen véleményt társítanak.

Mivel nincs egy kiforrott legjobb módszer, többfélét is elfogadottnak tartanak a kutatók, emellett kevés információ áll rendelkezésre az eljárások relatív teljesítményéről. Ritkán fordul elő, hogy több

adathalmazon vizsgálják meg, hogy egy új modell milyen relatív teljesítményt nyújt a korábbiakhoz képest. Sok esetben fekete dobozként használják a korábbi eljárásokat, és nem kérdőjelezik meg azok létjogosultságát, vagy megfelelőségét.

A sentibench kutatás során [3], a mondatok szintjén vizsgálják meg a különböző eljárások pontosságát. A fő kérdés a polaritás megállapítása, vagyis, hogy az egyes mondatokban pozitív, negatív, vagy neutrális-e a hangvétel. Különböző kiindulási feltételekkel vizsgálják meg az egyes eljárásokat, a vizsgálat egyik esetében különbséget tesznek a negatív és a pozitív polaritású mondatok között, míg a másikban háromféle polaritást különböztetnek meg. A kizárólag pozitív és negatív, kétosztályú mondatok között a modellek könnyebben döntenek, az accuracy, precision, Macro F1 metrika értékek is magasabbak. A Tweets_RND_II adathalmazon például a VADER modell 99,04%-os pontosságot ér el, míg a legrosszabbul a Stanford DM teljesít, 60,46%-kal. Mivel a méréseket sok adatbázissal elvégezték, átfogóbb képet kapunk, és kiderül, hogy ezek a metrikák erősen függenek attól, hogy melyik adathalmazzal dolgozik a modell. Az előbb említett VADER pl. három másik adathalmazon a vizsgált modellek közül a 11. 13. és 10. legjobb teljesítményű, pedig a Tweets_RND_II-n első volt.

Ha megnézzük a három osztályú analízis mérőszámait, amit összesen négy adathalmazon vizsgáltak, az accuracy értékek jelentősen alacsonyabbak. Például a Tweets_Semeval adathalmazon a legalacsonyabb érték 22,54%-os, míg a legmagasabb a LIWC15 modell 62,56%-os eredménye. A Tweets_RND_III adathalmazon a leghatékonyabb modell 76,4%-os accuracy értéket teljesít, ami a háromosztályú analízisben a legmagasabb az összes adathalmazon.

Megállapítható, hogy az akkori módszerek elfogadható eredményeket produkálnak, de jelentősen javíthatók. Az is kiderül, hogy a legtöbb módszer hatékonyabb a pozitív polaritás észlelésében, mint a negatív, vagy a neutrális esetekben. A legnehezebbnek a neutrális mondatok besorolása bizonyult. Az is kiderül, hogy a modellek jobban teljesítenek azokon az adathalmazokon, melyeken validálásra kerültek, ami a finomhangolás folyamatával magyarázható. Hogyha nem vesszük figyelembe ezeket az adathalmazokat, akkor az átlagos teljesítménye a validált modelleknek alacsonyabb lesz. Ezek mellett látható az eredményekből, hogy a módszerek hatékonysága az adatok forrása szerint is különbözik. Más-más modellek jobban teljesítenek a közösségi média üzeneteken, mint a kommenteken vagy a felhasználói visszajelzéseken.

2.6 BERT modell használata véleményanalízisre

Jelentős ugrás volt a véleményanalízis terén a BERT (Bidirectional Encoder Representations from Transformers, 2018) modell és annak finomhangolt változatainak használata, pl. (RoBERTa).

A BERT megjelenése előtt a legjobb módszerekkel 63,3%-os makro F1 értéket értek el a 2014-es évben kiadott SemEval twitteres adathalmazon [6]. A jobb érthetőség érdekében definiálom a különböző mérőszámokat:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

$$\text{makroF1} = \frac{\sum \text{F1 értékek}}{\text{Osztályok száma}} \tag{5}$$

Az 1. egyenletben a precision érték képlete található, melyet úgy kapunk, hogy a True Positive elemek számát elosztjuk a True Positive és a False Positive elemek számának összegével. A második egyenlet a recall értéket adja meg, hasonló módon a precision értékhez, a számlálóban TP van, nevezőben pedig TP+FN, ahol az FN a False Negative elemek számát jelenti. A 3. egyenletben a precision és a recall-ból származtatott F1 érték található, míg a 4. egyenlet megadja a makro F1 értékét, ahol a vesszük az összes F1 értéket és elosztjuk az osztályok számával.

A BERT előtt hagyományos gépi tanulásos módszereket és kézzel készített vélemény szótárakat használtak pl. SentiWordNet (2010) [7]. Ugyanezen a teszthalmazon a BERT alapmodellje, a BERT-SPC 73,6%-os makro F1 értéket ér el [6], míg további kiegészítő downstream architektúrával az LCF-BERT 75,8%-os F1 értékű [8]. Más munkák során a BERT modellt doménspecifikusan finom hangolják, illetve külső tudásbázist is felhasználnak, vélemény vagy érzelem szótárakat [9]. A GPT modell különböző változatai még hatékonyabbak az ilyen downstream feladatok során, mint a véleményanalízis, vagy más nyelvi feladatok, célom ennek a vizsgálata.

3 Célzott véleményanalízis a NewsSentiment adathalmazon

3.1 NewsSentiment adathalmaz bemutatása

A feladat során megismertem a véleményanalízis (sentiment analysis – SA), és a target-dependent sentiment classification (TSC) témakörét [10], melyen belül pedig egy többcélpontú klasszifikációs problémát oldottam meg, különböző megközelítésekkel. TSC során írott szövegekből természetes nyelvfeldolgozás segítségével állapítjuk meg az adott szöveg hangvételét, általános értelemben (pl. pozitív, negatív hangvétel, visszajelzés), vagy egy a szövegben lévő alanyon értelmezve.

A NewsSentiment [11] adathalmaz hosszú idő alatt készült, sok ember munkájával, arra a célra, hogy jó minőségű (TSC) analízist lehessen végezni rajta. Mind tanításra, mind tesztelésre fel lehet használni, 11300 mondat található benne, jellemzően több célponttal és a hozzájuk tartozó polaritásértékekkel.

Két másik adathalmazt is felhasználtak a NewsSentiment készítéséhez, az egyik a POLUSA, a másik a Bias Flipper 2018 (BF18). Mindkettő politikai cikkeket tartalmaz, és jól reprezentálja az online média kínálatát egy amerikai fogyasztó szemszögéből. Emellett sokféle témájú cikk megtalálható bennük, egy viszonylag széles időintervallumból. A POLUSA 0,9 millió cikket tartalmaz, melyek 2017 január és 2019 augusztus között íródtak, míg a BF18-ban 6447 cikk van, melyek összesen 2781 eseményhez köthetőek. Előnye a két halmaznak, hogy széles politikai spektrumot lefednek, egészen a szélsőbaloldalitól a szélsőjobboldali híradásokig. Emellett kevés szöveges hiba található bennük, annak ellenére, hogy főleg az internetről lettek crawling és scraping eljárásokkal, automatikusan kigyűjtve.

Tehát a klasszifikációt a NewsSentiment adathalmazon végeztem, melyben híradásokból, elektronikus hírekből kivett mondatok találhatók, címkézve és lektorálva. A munka finomhangolásos szakaszában az adatokat, mint tanító halmazt használtam fel. A NewsSentiment három különböző fájlból áll, egy 7758 mondatból álló címkézett „train” halmazból, amit tanításra használtam fel, a tanító halmaz mellett még két különálló fájlra oszlik meg, a devtest_rw és a devtest_mt-re, amelyeket tesztelésre használtam.

A szövegben a célpontokat különbözőféleképp kategorizálhatjuk, alapvető megkülönböztetés lehet a célpont pozitív, negatív megítélése, de lehet több kategóriával is dolgozni, a NewsSentiment adathalmazban három különböző kategóriát határoztak meg a szerzők, negatív, neutrális és pozitív kategóriát. A saját munkámban is ezt a kategorizálást használtam, annak érdekében, hogy a saját eredményeket össze lehessen vetni a kutatásban leírtakkal. Az adatok feldolgozásához többek között a BERT [12] nagy nyelvi modellt, illetve annak finomhangolt változatát a RoBERTa-t [13] használták, valamint egyedi kiegészítő architektúrát. Én a GPT-3 modellcsalád öt különböző típusát, az ada-001, a babbage-001, a curie-001, a davinci-003 és a gpt-3.5-turbo-t használtam. [14]

Az adatok saját feldolgozását a Google Colab környezetben végeztem, python nyelven, összevetettem a feldolgozás során a zero-shot, és a few-shot eljárással kapott eredményeket, valamint a fine-tune (finomhangolt) modellek teljesítményét. Valamint összevetem az eredményeket a 2021-ben íródott cikkben levőkkel, és az akkori „state of the art” modellek teljesítményével.

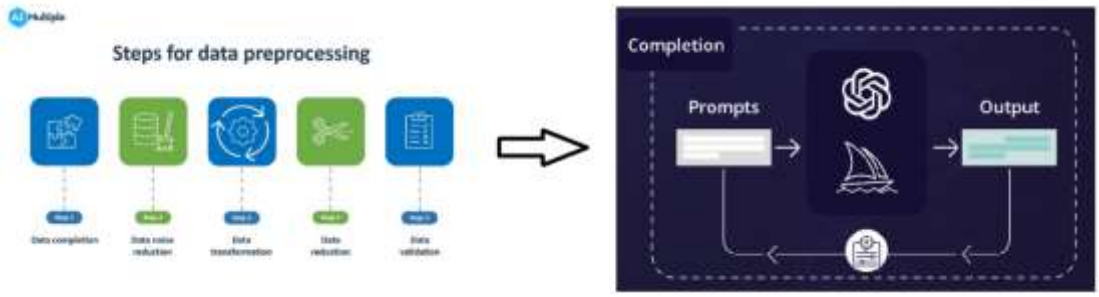
3.2 NewsSentiment adathalmazon elért state-of-the-art eredmények

A TSC módszerek kiértékeléséhez gyakran használják az alábbi három adathalmazt: Twitter [15], Laptop és Restaurant [16]. Mindegyiknél jellemző, hogy ritka az implicit kifejezése az érzelmeknek, az ilyen és hasonló, pl. közösségi média forrásoknál jellemzően explicit, egyértelmű módon fejezik ki a véleményüket a szerzők. Másik jellemzőjük, hogy nem foglalkoznak azzal, hogy a szövegben többféle vélemény is megjelenik, különböző polaritásokkal, és hogy több említése is előfordulhat az adott célpontoknak.

Az ilyen forrásoktól különböznek a híradásban közölt cikkek, ahol a szerzők jellemzően nem fejtik ki explicit a véleményüket, hanem törekednek az objektivitásra és csak közvetett módon utalnak a véleményre.

Tesztelés során a devtest_rw és a devtest_mt halmazt használták, és makro F1 értéket, Fpn (pozitív, negatív), accuracy értékeket használtak a modellek teljesítményének bemutatására. A devtest_rw halmazon a RoBERTa-GRU modellel 83,1%-os makro F1 értéket, 82,9%-os F1pn értéket, és 83,8%-os accuracyt mértek. A devtest_mt adathalmazon $F1_m$=82,5%, $F1_{pn}$= 80,2%, accuracy=84,6% értékek lettek.

3.3 Adatok beolvasása, API-ok használata



3. Ábra: Folyamatábra a munka lépéseiről [4,5]

A harmadik ábrán látszanak az adatfeldolgozás és a prompt engineering folyamat lépései. Az első lépés a szöveges adatok megfelelő feldolgozása volt. Az eredeti adathalmaz jsonl formátumban található, és három különböző fájlban van tárolva. Az adatok nagy részét egy tanítóhalmaz teszi ki, ebben a fájlban 7758 mondat található, fel vannak sorolva a különböző alanyok (targetek) és azok pozíciója, a hozzájuk tartozó polaritásértékek (2, 4, 6), az alanyok későbbi említései a mondatokban. A 2-es polaritásérték a negatív véleményt jelenti, a 4-es a neutrálist és a 6-os a pozitívot. A másik két fájl, két teszthalmaz, a devtest_mt és a devtest_rw. A devtest_mt fájlban minden mondathoz legalább két alany tartozik, míg a devtest_rw fájlban nincs ilyen megkötés. A devtest_rw „rw" része a „real world" -re utal, vagyis hasonló a valós példák előfordulásához a különböző híradási cikkekben, míg az „mt" a multi-target jelző.

Az adatokat json formátumból pandas dataframe-be olvastam, hogy könnyeb legyen a további feldolgozás. A tanító halmazban minden mondathoz tartozik legalább egy target, vagyis alany, és annak polaritása, azonban az devtest_mt esetben 7758 példányhoz tartozik két alany, hét esetben három alany, és egy esetben négy alany is. Emellett a különböző célpontokhoz további említések is tartozhatnak (further mentions). Ilyen példa az adathalmazban, (mention-Hillary Clinton) bejegyzéspár, és (further mention-she) rekordok. Ezek a further mention említések, lehetnek

[4] https://research.aimultiple.com/data-preprocessing/

[5] https://thepraxiz.com/prompt-engineering-everything-to-know/

különbözőek, vagy megegyezőek is az eredeti célponttal és egymással is, (a polaritásértékeik viszont megegyeznek az eredeti célpont polaritásával). Ezekkel a további említésekkel azonban nem foglalkoztam a feldolgozás során, a NewsSentiment kiértékelésekor sem használták fel őket, és a kigyűjtésük automatikusan történt.

Miután megvizsgáltam az adatokat, saját feldolgozásra újra rendszereztem őket, külön-külön oszlopokba kigyűjtve az egyes példa mondatokat, a különböző alanyokat, további említéseket, polaritásértékeket stb. Erre azért volt szükség, mert a json formátumban különböző mezőkben és almezőkben voltak tárolva a szükséges adatok, (pl. az alany és a hozzá tartozó polaritásérték) és így nem lehetett egyszerűen promptokat generálni belőlük. Prompt alatt olyan, jellemzően angol nyelvű szöveget értünk, amelynek segítségével egy nagy nyelvi modellel kommunikálni lehet, annak érdekében, hogy az elvárt kimenetet generálja. Ez a szöveg lehet például kérdő mondat, kódrészlet, vagy egy megadott példa formájában is.

Az adatok elemzéséhez a GPT-3 modellcsaládot használtam fel, azon belül pedig a text-ada-001, a text-babbage-001, a text-curie-001, a text-davinci-003 és a ChatGPT (GPT-3.5-turbo) modellt. Ezeket a modelleket az OpenAi cég fejlesztette ki, és a GPT szériában a harmadik iterációhoz tartoznak. Az ada a legkisebb, leggyorsabb modell, míg a davinci a legnagyobb. Ezek mellett az ada és a curie modellek általam finomhangolt változatait is leteszteltem.

Ehhez a modellcsaládhoz hozzáférni egy API-on keresztül lehet, bizonyos korlátozott keretek között. Ilyen korlát, többek között a rate limit, illetve a felhasználó rendelkezésére álló kreditek. Az általános rate limit, vagyis, hogy hány kérést indíthatunk a modellek felé percenként, ingyenes felhasználás esetén, a GPT-3 modelleknél 60/perc, míg a ChatGPT esetében 3/perc. A betáplált információ és a kijövő információ összes mennyisége is meg van határozva, ez a GPT-3 modellek esetében[6] 2049 token, míg a davinci-003-nál és a GPT-3.5-turbonál 4097 token. Fontos megjegyezni, hogy egy token közelítőleg 0,75 angol nyelvű szónak felelnek meg[7]. Az API felhasználó ingyenesen öt dollárnyi kreditet kap, és a modellek között a felhasználás költsége nem egyenletes, legolcsóbb az ada modell esetében, míg a legdrágább a davinci modellnél egy tokenre számolva.

[6] https://platform.openai.com/docs/models/gpt-3-5

[7] https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

Modell	InstructGPT-$/1000 token	Fine-tuning models (training) $/1000 token	Fine-tuning models (usage) $/1000 token	Max Tokens (darab)
Ada	0,0004	0,0004	0,0016	2049
Babbage	0,0005	0,0006	0,0024	2049
Curie	0,0020	0,0030	0,0120	2049
Davinci	0,0200	0,0300	0,1200	4097
GPT-3.5-turbo	0,0020	-	-	4096

2. Táblázat: A GPT-3 modellek használati költségei, Instruct és Fine-Tuning szempontjából, és a maximális token számok

A második táblázat összefoglalja minden GPT-3 variáns használati költségeit, kezdve az ada modellel, amely összesítésben a legolcsóbb, egészen a legdrágább davinci modellig. A táblázatban emellett még szerepel az összesen kommunikálható maximális ki és bemeneti tokenek száma.

Az API-ok használatához szükséges egy felhasználónként egyedi API kulcs, és az utasításokhoz részletes útmutatók[8] találhatók. Számos paraméter[9] adható meg az API hívások során, ilyenek például a „temperature”, a „maxtoken”, „maximum length”, a „top P”, a „frequency penalty” és a „presence penalty”. Ezeknél a modelleknél a két legfontosabb paraméter a temperature, amely 0 és 2 között mozoghat és a generált szöveg válaszának véletlenszerűségét kontrollálja, valamint a maxtoken paraméter, amely a modell válaszának maximális tokenszámát adja meg. A temperature esetén a 0 érték determinisztikus és repetitív válaszokat, míg a magasabb értékek egyre véletlenszerűbb válaszokat jelentenek. A „maximum length”-el lehet megadni, hogy a prompt és a kimenet együtt hány token lehet, ez azonban modellfüggő, a gpt-3.5-turbo esetén pl. ez minimum 1, maximum 4096. A Top P egy 0 és 1 közötti érték, ami a válasz diverzitását kontrollálja mintavételezéssel. A frequency penalty 0 és 2 között változhat, és azt befolyásolja, hogy egy új token esetén mekkora legyen a büntetés érték a token korábbi előfordulási gyakorisága alapján. Ha az érték nagy, a modell jobban

[8] https://platform.openai.com/docs/guides/fine-tuning

[9] https://platform.openai.com/docs/api-reference/completions/create

kerüli az ismétléseket a válaszában. A presence penalty pedig hasonlóan a frequency penaltyhoz egy büntetésértéket definiál 0 és 2 között, ami a token ismétlések kikerülésére való, így a modell nagyobb valószínűséggel fog új témáról beszélni. Munkám során a két legfontosabb paraméter, a temperature és maxtoken értékét változtattam, minden más az alapértelmezett értékek szerint került beállításra.

3.4 Elvégzett prompt engineering bemutatása

Munkám egy részében optimális promptokat kerestem, prompt engineering-et végeztem, vagyis olyan bemeneti szöveget kerestem a modellek számára, ahol az általam elvárt kimenet minősége a legjobb lett. A minőséget pontosság értékkel vizsgáltam, vagyis összevetettem a már címkézett NewsSentiment adathalmaz polarity értékeivel a modellek válaszait és kiszámoltam a helyes válaszok arányát. Optimális esetben a modell bemenete a példamondat, némi kiegészítő szöveg, ami utal a feladat milyenségére, vagyis, hogy melyik célpontot vizsgálom véleményanalízis szempontjából, és esetleg az elvárandó kategóriák felsorolása (pozitív, negatív, neutrális). Kimenetként egyszavas (egy tokenes) választ várok, ami helyes analízis esetében megegyezik az adott mondathoz tartozó polaritással. Egy példaprompt így nézett ki:

Prompt:

„Is the sentiment towards … as target, positive, negative or neutral in the following sentence?”

A példában a … helyén a célpont, mint változó állt, és a kérdőjel után a kérdéses mondat szerepelt. A kipróbált promptok különböznek a hosszukban, a szóhasználatukban a szórendjükben, abban, hogy szerepel-e bennük a lehetséges válaszok felsorolása. Minden promptot kis példaszámmal vizsgáltam először, jellemzően tíz példamondattal, majd, ha értelmezhető válaszokat adtak a modellek, akkor egy, 100 mintából álló teszthalmazra is kipróbáltam.

Az előzetes vizsgálat után gyakran elvetettem promptokat, lehetőség szerint próbáltam a pontosabb válaszokat adókkal tovább dolgozni. Ez azt jelenti, hogy megnéztem egy adott prompttal a modell tíz példamondatból hány esetben ad helyes választ az alany polaritására. Ha ez négynél kevesebb helyes választ adott, elvetettem a promptot és másikkal próbálkoztam tovább. Ha ez az érték magasabb volt, mint négy, 100 példán is ellenőriztem a promptra adott válaszokat.

A legjobb promptok zero-shot esetben jellemzően 5-6 helyes választ is adtak, ez azonban függött attól, hogy melyik modellt vizsgáltam. A kisebb modellek esetében (ada, babbage) előfordult, hogy csak 3-4 helyes választ adtak a tízből, ilyen esetben azonban nem vetettem el a promptot, 100 példára is lefuttattam, mivel összességében kevesebb jó választ vártam.

Erre a szelekcióra többek között azért volt szükség, mert az időm és a felhasználható krediteim mennyisége is véges volt, vagyis bele kellett férjek a pénzügyi keretbe. A dolgozatban nem sorolom fel az összes mérési eredményt, összesen 13 különböző prompttal végeztem 100 példára accuracy mérést, a finomhangolt modellek esetén 200, 500, 500, 500 példára és a legjobb finomhangolt modell esetében a teljes teszthalmaz 1146 és 1476 alanyára megnéztem a pontosságot. A teljes vizsgálat során egy táblázatot készítettem, amelyben összefoglaltam az összes eredményt, a modellek nevét, a paraméterek értékeit (temperature és maxtoken), a felhasznált promptokat, a tízes és százas accuracy értéket, több esetben is a modell teljes válaszát, megjegyzéseket és a legjobb modellek esetében minden példa vizsgálatának eredményét külön elmentve.

Továbbá megvizsgáltam, azt az esetet, amikor a maximális tokenszámot kihasználva, annyi mondatot adtam meg a modellnek bemenetként amennyi belefért a tokenkeretbe, és egyetlen sor utasítást adtam arra, hogy végezzen TSC-t a különböző alanyokra vonatkozóan. Ehhez egy teljesen más, hosszú bemenő promptot használtam, és az eredmények azt mutatták, hogy pl. a Chat-GPT 64%-os pontossággal jó választ adott. Ez az érték mindössze 5%-kal rosszabb a zero-shot eljárásban külön-külön beadott példák esetében mért pontosságnál. Így levonható konklúzióként, hogy akár gyorsan, egyszerre is lehet sok adattal TSC-t végzeni, anélkül, hogy külön hívásokat végeznénk a Chat-GPT-vel, azonban ezek pontossága rosszabb.

Háromféle eljárás szerint végeztem a vizsgálatokat, zero-shot, few-shot (10 példa), illetve fine-tune esetben, és a mérések few-shot és zero-shot eredményét az alábbi táblázatban foglaltam össze.

Modell	Módszer	Temperature	Tokenszám	Accuracy
Babbage	Zero-shot	0	alapértelmezett	**62%**
Curie	Zero-shot	0,7	4	**51%**
Davinci	Zero-shot	0	alapértelmezett	**67%**
GPT-3.5-turbo	Zero-shot	0	alapértelmezett	**69%**
Ada	Few-shot	0	alapértelmezett	**43%**
Babbage	Few-shot	0	alapértelmezett	-
Curie	Few-shot	0	alapértelmezett	**62%**
Davinci	Few-shot	0	3	**67%**
GPT-3.5-turbo	Few-shot	0	alapértelmezett	**53%**

3. Táblázat: GPT-3 modellek és ChatGPT modell zero-shot és few-shot paraméterei

A 3. táblázatról leolvasható, a modellek teljesítménye a zero-shot módszerrel. A babbage modell a legjobb prompttal 62%-os, a curie 51%-os, a davinci 67%-os, a gpt-3.5-turbo 69%-os pontosságú volt. A legjobb eredményt zero-shot esetben a ChatGPT adta, ami egyben a legnagyobb modell is a felsoroltak közül, így ez nem okozott meglepetést. A temperature paraméter értékét nullának választottam, ugyanis ez indokolt klasszifikációs esetben, ahol nem szükséges, hogy futásonként különböző válaszokat generáljon egy modell. A működés vizsgálata érdekében kipróbáltam 0,7-es értékkel is, azonban ez a legtöbb esetben nem okozott jelentős változást a kimenetek minőségében, így minden további esetben nullát használtam temperature értéknek.

A másik fontos paraméter a válasz maximális tokenjeienek száma volt, itt az elvárt egy tokenes választás jellemzően nem adott értelmezhető választ. A legtöbb esetben üres tokenes, vagy véletlenszerű válaszokat generáltak a modellek, viszont a davinci modell és a chatgpt válaszai értelmezhetőek voltak. Több tokenszám értéket is kipróbáltam, 2-3-4-5-10-20, illetve az alapértelmezett 2049 (GPT-3) illetve 4097 (GPT-3.5-turbo) értékeket is. Jellemzően a modellek nagyobb tokenszám esetén jobban teljesítettek, itt azonban fontos volt kiszűrni az olyan válaszokat, ahol a modell kimenetében többféle polaritás is szerepelt. Ezeket a válaszokat ellenőriztem, és csak azokat fogadtam el helyesnek, ahol egyetlen polaritás szerepelt a válaszban.

Few-shot esetben először egy példát adtam meg a modelleknek, majd tíz példával vizsgáltam a kimeneteket. Jellemzően az egy példamondat-példakimenet pár hozzáadása a promptokhoz rosszabb

eredményt adott a zero-shot esetnél. Tíz különböző példa bemenet-kimenet párt megadva a pontosság érték egyes esetekben javult (pl. Curie modell), más modelleknél romlott pl. Chat-GPT, a davinci modell esetében ugyanúgy 67%-os lett. Ez a véletlenszerű változás magyarázható a [5] cikkben vizsgált modellek viselkedésével. A kutatás eredményéből látszik az, hogy a példák számának emelésével a modellek pontossága az adott feladat terén általában nő, emellett azt látjuk, hogy általában a nagyobb modellek jobban teljesítenek in-context learning feladatokban. Ezek a következtetések viszont csak egy összegzett eredményt mutatnak, ettől jelentősen eltérő esetek is előfordulhatnak, ahol a modellek adott kontextusban rosszabban teljesítenek, mint zero-shot esetben.

3.5 Ada és Curie modellek finomhangolása

Ahhoz, hogy pontosabb véleményanalízist végezzek, finom hangolni kezdtem a különböző kiindulási modelleket. Ezt az anaconda környezetben végeztem, az openai honlapon talált fine-tune leírások alapján.

Az OpenAI API biztosít egy CLI programot, amelyben az adatok előkészítését, szerkesztését el lehet végezni. A program bemenetként pl. xlsx, vagy json formátumú fájlokat vár, melyekben szerepelnie kell egy prompt és egy válasz (completion) mezőnek. Ez a fájl lesz a tanító adathalmaza a kiindulási modellnek. A modell tanításához használt fájl jsonl formátumú kell legyen, amit a programmal a tanító fájlt konvertálva elő lehet állítani. A program azt javasolja, hogy távolítsuk el a promptok elején lévő ismétlődő részeket, mivel ezek nem szükségesek a finomhangolás során, a few-shot esettel ellentétben. Továbbá a prompt végére érdemes lezáró szekvenciát tenni, ami jelzi a modellnek, hogy véget ért a bemenet, a completion részt pedig üres (whitespace) karakterrel ajánlott kezdeni. Emellett a program segítségével eltávolíthatók az ismétlődő sorok az adatokban, ebből a feldolgozásakor nyolc darab volt, és fel lehet bontani az adatokat tanító és validációs halmazokra is.

A tanítófájl előkészítése után az feltöltésre kerül, majd egy paranccsal megadható, hogy melyik fájl alapján végezzünk finomhangolást, melyik kiindulási modellen. Ennek költsége és időigénye változó, a program a költséget megadja, az időtartamra pedig becslést ad. A tanítás költsége kisebb modellek esetén alacsonyabb, míg a nagyobb modellek esetén ennek tízszerese is lehet. A munkafolyamat ezután várólistára kerül és amennyiben rendelkezésünkre áll a megfelelő

kreditmennyiség, a tanítás megkezdődik. A Curie modell 800 példával való finomhangolásakor, az alapértelmezett paraméterek szerint, a tanítás 4 epochig tartott, a batch_size 1 volt, learning_rate_multiplier 0,1 értékű, prompt_loss_weight 0,01. Az alapértelmezett értékeket természetesen lehet változtatni. A finomhangolás aktuális állapotát le lehet kérdezni, és meg is lehet szakítani menet közben a folyamatot.

A tanítás végén visszakapunk egy egyedi azonosítóval (például időbélyeggel) ellátott finomhangolt modellt, amivel aztán az API-n keresztül tudunk kommunikálni.

A finomhangolás során kimerítettem az általam elérhető krediteket. Ingyenes felhasználás esetén egy modell tanítása maximum öt dollár költségű lehet, ami a curie modell esetében 7400 mondatnyi bemenetet jelentett, vagyis majdnem a teljes tanítóhalmazt. Ennek a tanításnak a költsége a davinci modellt esetén tízszer ennyibe került volna, ami nem volt elérhető számomra. Mivel a davinci modell minden korábbi bemeneten jobban teljesített a kisebbik curie modellnél, feltehető, hogy még jobb eredményt adott volna a finomhangolás után. Ehhez azonban további vizsgálatok, és források szükségesek.

3.6 Eredmények összevetése

A [11] cikk esetében a két vizsgált teszt halmazon (devtest_mt, illetve devtest_rw) elért legjobb eredményeket összevetem a finomhangolás eredményével. Mindkét teszt halmazon a legnagyobb accuracy értéket a RoBERTa finomhangolt modell GRU változatával érték el a szerzők, a devtest_mt esetében 84,6%-ot, és a devtest_rw esetében 83,8%-ot.

Összesen négy különböző finomhangolást végeztem, az első a curie modellen történt, a koncepció kipróbálására, 800 bemeneti példával, és az accuracy mérésére 200 példát használtam fel. Ennél a tanításnál a teljes promptot felhasználtam bemenetként, nem lett levágva a mondatok eleje, illetve a stop sequence sem került alkalmazásra, a whitespace sem került be a válaszok elejére. Ezt a modellt 200 tesztpéldán megvizsgáltam, ahol 74,5% pontosságot ért el, amennyiben egy tokenes választ vártam.

Másodszor a finomhangolás alap modelljének az ada-001-et választottam, melynek tanítási költsége sokkal kisebb a curie-éhez képest (ada 0,0004 $/1000 token, míg a curie 0,003$/1000 token),

és bemenetként 6750 példamondatot adtam meg. A tanítóhalmaz 7750 mondatából fennmaradó 1000 példamondatot használtam fel tesztelésre. A bemeneti szöveg szerkesztését az ajánlott megfontolások szerint végeztem. Ennek a kész modellnek a pontosabb vizsgálatához az 1000 példamondatból 500 ellenőrző példán végeztem mérést. A válaszok pontossága 81,4%-os volt, ami megközelíti a RoBERTa-GRU modell teljesítményét, és emiatt a jó eredmény miatt további finomhangolásokat végeztem.

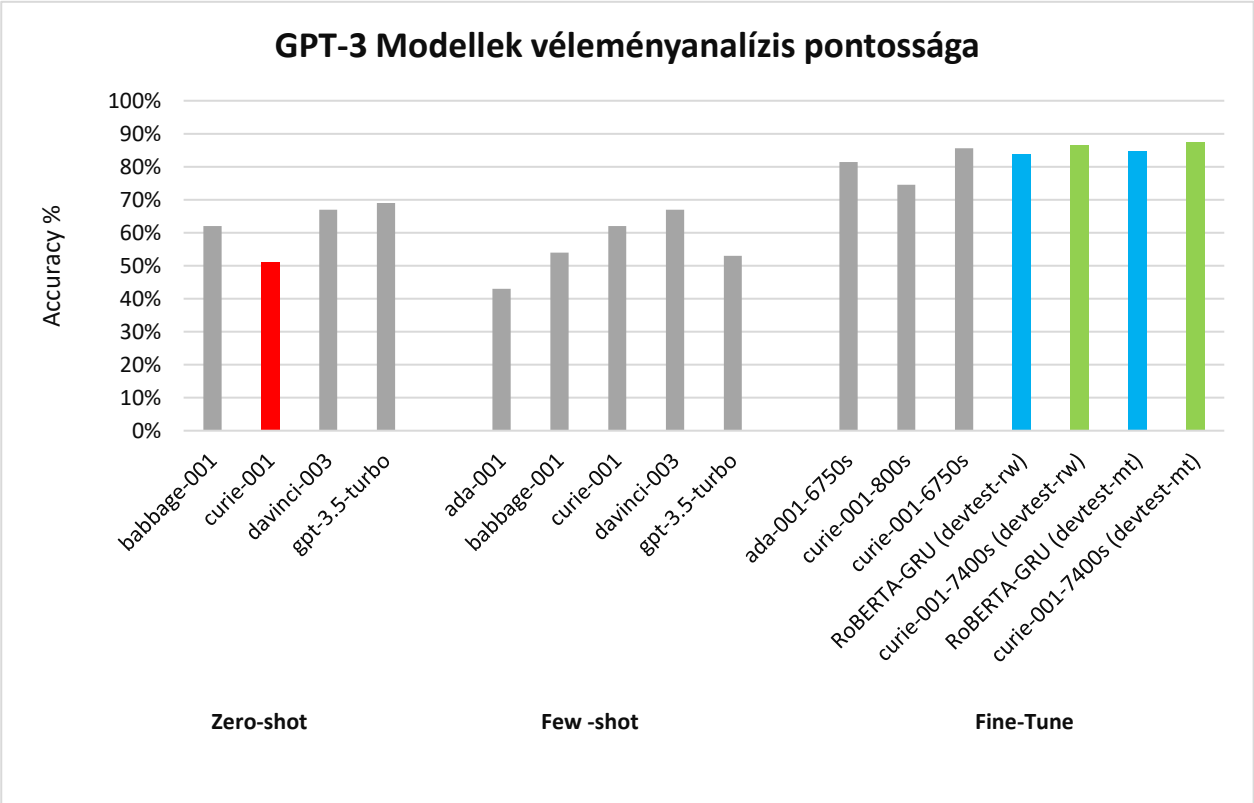
Harmadik esetben a finomhangolás alapjául ismét a curie modellt választottam, mivel ez volt a legnagyobb modell, amelyet még képes voltam az adott költségkeretek között tanítani. A tanításhoz 6750 példamondatot használtam, és mivel a korábbi eredmények javultak a promptok ajánlott szerkesztésével, ezt ismét elvégeztem. A kész modell pontosságát ugyanazon az 500 példán teszteltem, két különböző maxtoken paraméter értékkel. A két tokenes válasz esetében 84,8%-os pontosságot mértem, míg az egy tokenes válasz esetében ez 85,6% volt. Ezek az értékek meghaladják a RoBERTa-GRU modell pontosságát, azonban nem azon a teszthalmazon mértem őket, mint amit a modellnél használtak, így az összehasonlítás nem lehetséges.

Végül a curie modellt felhasználva és a maximális tanítási költségkereten belül maradva végeztem finomhangolást, 7400 példamondattal (csaknem a teljes tanító halmaz a NewsSentiment esetében). Itt is a tanító API által ajánlott formájúra szerkesztettem a promptokat, és a modell teljesítményét most a devtest_rw illetve a devtest_mt halmazokon mértem. A finomhangolt curie modell pontossága a devtest_rw esetében 86,47%-os lett, ami 2,6%-kal meghaladja a RoBERTa-GRU teljesítményét, a devtest_mt esetében pedig 87,33%-os lett, ami 2,7%-kal meghaladja azt.

Modell	Módszer	Temperature	Max Tokens (darab)	Accuracy (500 példa)	Accuracy devtest_rw	Accuracy devtest_mt
RoBERTA-GRU	-	-	-	-	**83,8%**	**84,6%**
Curie-FT	Fine-Tune 800 példával	0	1	**74,5%**	-	-
Ada-FT	Fine-Tune 6750 példával	0	1	**81,4%**	-	-
Curie-FT	Fine-Tune 6750 példával	0	2	**84,8%**	-	-
Curie-FT	Fine-Tune 7400 példával	0	1	-	**86,47%**	**87,33%**

4. Táblázat: RoBERTa-GRU modell accuracy eredményei, finomhangolt Ada és Curie modellek accuracy eredményei

A 4. táblázatban összefoglalva szerepelnek a különböző finomhangolt modellek adatai. Első oszlopban a modell neve szerepel, a második oszlop a tanító példaszámra utal, ezt követi a temperature érték, majd a maximális tokenszám, végül az utolsó három oszlopban az egyes teszt részhalmazokon mért pontosság értékek vannak.



4. Ábra: GPT-3 Modellek zero-shot, few-shot és fine-tune pontossága

A 4. ábrán balról-jobbra haladva látszanak a modellek adott eljárásban elért legjobb pontosságértékei. Először felsoroltam a zero-shot mérés eredményeit, a babbage-001, curie-001, davinci-001, gpt-3.5-turbo modellek esetében, majd ezeket követik a few-shot eredmények, az ada-001, és a korábban felsorolt modellek pontosságával, és végül a fine-tune eredmények látszanak, az ada-001-6750s (ada-001 modell, 6750 mondattal tanított verzió), curie-001-800s stb. A legjobban teljesítő finomhangolt modell esetében (curie-001-7400s) mind a két teszthalmazon (devtest-rw

illetve devtest-mt), mért pontosságot feltüntetem, ezek mellet szerepel a RoBERTA-GRU modell ugyanezen halmazokon mért pontossága.

Az ábrán pirossal jelölt oszlop a curie-001 modell 51%-os zero-shot pontosságát mutatja. Ez az érték 35,47%-kal alacsonyabb az általam finomhangolt, zöld oszloppal megjelenített curie001-7400s modell (devtest-rw) halmazon mért 86,47%-os pontosságánál. Valamint ez a zero-shot érték 36,33%-kal alacsonyabb a szintén zöld oszloppal jelölt, curie001-7400s, (devtest-mt) 87,33%-os pontosságánál.

A kék oszlopok, a RoBERTA-GRU modell 83,8%-os és 84,6%-os pontosságát mutatják, amit a devtest-rw, és a devtes-mt halmazon mértek a NewsSentiment kutatásban. Az eredményekből látszik, hogy a zero-shot eljáráshoz képest 35,47%-36,33%-kal pontosabb a finomhangolt modell teljesítménye.

3.7 Következtetések

Az útmutató [10] és a tapasztalataim alapján a finomhangolt modellek teljesítménye folyamatosan javult, minél nagyobb volt a tanítóhalmaz, illetve minél nagyobb méretű modellt használtam. Ezek alapján felteszem, hogy ha a teljes tanítóhalmazt (7758 mondat), illetve a davinci modellt használom kiindulásként még jobb pontosságot lehet elérni ezen a TSC feladaton. Amennyiben további adathalmazokat is felhasználnék, pl. Twitter, Laptop, Restaurant stb. véleményem szerint lehetséges lenne tovább javítani az eredményeket, és még jobb véleményanalízist végezni.

Érdemes lenne tovább kísérletezni a davinci modellel, illetve a GPT-4 modellel, zero-shot módszerrel, és finomhangolással is, a GPT-4-nek azonban jelenleg korlátozott és a használt modelleknél drágább a hozzáférése, finomhangolása nem is végezhető.

[10] https://platform.openai.com/docs/guides/fine-tuning

4 Összefoglalás

Ebben a dolgozatban a TSC, target-dependent sentiment classification problémakörrel foglalkoztam. A probléma többféle formában, egy és többcélpontú elemzés, különböző kategóriájú klasszifikációk formájában, is előkerül az NLP gyakorlatban. Ilyen alkalmazás lehet nagyméretű szövegek feldolgozása, review-k kategorizálása, tőzsdei elemzés, hírek vizsgálata.

A téma irodalmának vizsgálata után, a GPT-3 modellcsalád elemeivel dolgoztam, először zero-shot, majd few-shot módszerrel, végül finomhangoltam az ada és a curie modelleket különböző méretű tanítóhalmazokkal, és különbözően összeállított példapromptokkal.

Az adatok vizsgálatát a colab és az anaconda környezetben végeztem, python nyelven. Felhasználtam a NewsSentiment adathalmaz, ami egy emberi munkával címkézett és lektorált halmaz. Az eredményeket összevetettem a NewsSentiment szerzőinek 2021-ben végzett vizsgálatával, ahol többek között a BERT modell RoBERTa finomhangolt modelljének módosításával értek el. A GPT-3 ada és curie modelljeinek finomhangolásával megközelítőleg hasonló eredményt lehet elérni, mint a RoBERTa-GRU modell esetében, illetve a Curie modell további tanításával 2,6%-kal jobb eredményt is.

Következtetésként le lehet vonni, hogy a különböző transzformer LLM modellek méretüktől és korszerűségüktől függően képesek rendkívül jó véleményanalízisre, ami magas pontosságot ér el lektorált adathalmazokon. A kisebb ada modell finomhangoláskor ugyanazon a tanítóhalmazon 4,2%-kal kisebb pontossággal válaszolt, mint a curie modell, míg a curie modell tanítópéldáinak 800-ról 7400-ra növelésével, illetve a tanítószöveg formázásával összesen 11,97%-kal jobb accuracy érték volt mérhető.

Irodalomjegyzék

[1] Lopez-Lira, A., & Tang, Y. (2023). Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models. *arXiv preprint arXiv:2304.07619*.

[2] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., & Stoyanov, V. (2019). SemEval-2016 task 4: Sentiment analysis in Twitter. *arXiv preprint arXiv:1912.01973*.

[3] Ribeiro, F. N., Araújo, M., Gonçalves, P., André Gonçalves, M., & Benevenuto, F. (2016). Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, *5*, 1-29.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[5] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901.

[6] Kiritchenko, S., Zhu, X., Cherry, C., & Mohammad, S. (2014, August). NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)* (pp. 437-442).

[7] Baccianella, S., Esuli, A., & Sebastiani, F. (2010, May). Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec* (Vol. 10, No. 2010, pp. 2200-2204).

[8] Zeng, B., Yang, H., Xu, R., Zhou, W., & Han, X. (2019). Lcf: A local context focus mechanism for aspect-based sentiment classification. *Applied Sciences*, *9*(16), 3389.

[9] Hosseinia, M., Dragut, E., & Mukherjee, A. (2020). Stance prediction for contemporary issues: Data and experiments. *arXiv preprint arXiv:2006.00052*.

[10] Hamborg, F., Donnay, K., & Gipp, B. (2021). Towards target-dependent sentiment classification in news articles. In *Diversity, Divergence, Dialogue: 16th International Conference, iConference 2021, Beijing, China, March 17–31, 2021, Proceedings, Part II 16* (pp. 156-166). Springer International Publishing.

[11] Hamborg, F., Donnay, K., & Merlo, P. (2021, April). NewsMTSC: a dataset for (multi-) target-dependent sentiment classification in political news articles. Association for Computational Linguistics (ACL).

[12] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[13] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[14] Dale, R. (2021). GPT-3: What's it good for?. *Natural Language Engineering*, *27*(1), 113-118.

[15] Manandhar, S., & Yuret, D. (2013, June). Second joint conference on lexical and computational semantics (* sem), volume 2: Proceedings of the seventh international workshop on semantic evaluation (semeval 2013). In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*.

[16] Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., ... & Eryiğit, G. (2016). Semeval-2016 task 5: Aspect based sentiment analysis. In *ProWorkshop on Semantic Evaluation (SemEval-2016)* (pp. 19-30). Association for Computational Linguistics.