TDK-dolgozat

2024.

Füri Erika Rebeka
Pénzügyi és Számviteli Kar

TERMÉKEK KATEGORIZÁLÁSA A BERT MODELL SEGÍTSÉGÉVEL

CATEGORIZATION OF PRODUCTS USING THE BERT MODEL

Dr. Kovács Endre

Kézirat lezárásának dátuma: 2024, November, 04

TARTALOM JEGYZÉK

TÁBLÁZATOK JEGYZÉKE

ÁBRA JEGYZÉK

1 BEVEZETÉS

Jelenleg negyedéves gazdaságinformatikus hallgató vagyok a Budapesti Gazdasági Egyetemen, ahol tanulmányaim során mélyebb betekintést nyerhettem az adatok kezelésének és feldolgozásának fontosságába. Megismerkedtem különböző módszerekkel és technikákkal, amelyek segítenek az adatelemzésben és az adatokból tanuló algoritmusok használatában, beleértve a mesterséges intelligenciát (MI) is. Mivel jelenleg egyre nagyobb népszerűségnek örvend ez utóbbi terület, így szerettem volna mélyebben beleásni magam, több tapasztalatot szerezni ebben a témában. Célom ezzel a dolgozattal nem csak egy feladat megvalósítása volt, hanem az is, hogy részese legyek egy ilyen volumenű projektnek, amelyet akár munkakeresés szempontjából is fel tudok használni.Manapság egyértelműen látszódik milyen népszerű ez az ágazat, egyre több vállalat próbál ezzel kapcsolatos fejlesztéseket beépíteni a saját rendszerébe, amely vagy kivált egy-egy ember által is fáradalmas feladatot, vagy épp segít a feladat megvalósításában. Dolgozatom lényege is, hogy egy webárúház munkafolyamatait segítsem, egy MI modell felállításával. Ez a modell a vállalat kategorizálási feladataiban fog segíteni a jövőben, amelynek segítségével könnyebben, gyorsabban és pontosabban fognak tudni bekategorizálni egy-egy terméket.

1.1 A KUTATÁS CÉLJA

Napjainkban egyre gyakrabban előfordul, hogy mesterséges intelligenciával próbálunk megoldani valami egyszerű feladatot. Vehetjük például egy novella megírását, amit a gimnáziumban kapott egy diák házi feladatnak, vagy épp egy egyszerű program alapjainak a megalkotását. Nagyvállalati szinten is egyre népszerűbb ezeknek a rendszereknek az alkalmazása, nagyon sokan használják például hivatalos e-mail-ek megfogalmazásának segítségében. Ez a tanulmány is az MI-ról szól, pontosabban egy adott modellt, a BERT-et (Bidirectional Encoder Representations from Transformers) fogom implementálni, tesztelni és az általa generált eredményeket elemezni.

Feladatként kaptam azt, hogy segítsek egy műszaki cikkeket áruló webes áruháznak létrehozni a modell segítségével egy olyan rendszert, ami képes pusztán név alapján kategorizálni az új termékeiket. A cég több évtizede van jelen Magyarországon, kezdetben családi vállalkozásként indult, viszont az évek során egyre jobban terjeszkedtek, és alkalmazottai száma mára már 50 és 99 fő között mozog. Hivatalos adat alapján a nettó árbevétele 2023-ban több, mint 8,5 milliárd forint volt, illetve a több mint 45 ezer szerszám internetes értékesítése mellett, olyan szolgáltatásokkal is foglalkozik, mint a szervizelés, gyártás és gépkölcsönzés. 3 boltot is vezet Hajdú-Bihar vármegyében, illetve idén

októberében megnyitott a 4. üzletük is. A vállalat mind amellett, hogy jelen van az EMAG oldalán is, az Árukereső internetes weboldalon elérte a „Megbízható Bolt” kitüntetését is. Ez utóbbi kitüntetést csak azok a boltok kaphatják meg, akiről a már ténylegesen vásárolt felhasználók értékelései alapján legalább 10 vélemény értékelése pozitív volt.

Ezen adatok alapján látszódik, hogy Magyarország egyik piacvezető vállalatának volt fontos egy megfelelő kategorizálási rendszert kiépíteni, melyet eddig manuálisan csinálták. Így, nagyjából 10%-os hibahatárral dolgoztak, illetve maga a folyamat is időigényes volt. A célom, hogy egy olyan modellt készítsek, amellyel a cég életét segítem azáltal, hogy ezt automatizálni tudják, illetve, hogy ezt a hibahatárt le tudjam csökkenteni közel a 0%-hoz.

Olyan kérdésekre keresem a választ, mint hogy milyen fontosabb követelmények kellenek ahhoz, hogy egy mesterséges intelligenciát használó modellt létre lehessen hozni, mi a folyamata, milyen lehetőségek rejlenek benne, illetve mik a gyengeségei.

1.2 KUTATÁSI KÉRDÉSEK

A kutatás során több kérdésem is felmerült a projekttel kapcsolatban. Ezek közül a fontosabbak a következők:

1. Tényleg megvalósítható-e, hogy egy BERT modellt be lehessen tanítani termékek kategorizálására, ezzel kiváltva az emberi manuális folyamatot?
2. Helyesen fog-e osztályozni, kevesebb hibával dolgozik, mint az emberi manuális kategorizálás? Milyen mértékben képes csökkenteni a manuális kategorizálási hibaarányt?
3. Működni fog-e egyáltalán a modell ekkora csoportosítási halmazzal, és a megfelelő csoportokba fogja-e besorolni a termékeket?
4. Jól fog-e teljesíteni a magyar nyelvkörnyezetben?
5. Erőforrás szempontjából lehetséges-e a megvalósítás? Milyen technikai és adatfeldolgozási követelményekre van szükség ahhoz, hogy a BERT modellt használni lehessen egy vállalati környezetben?
6. Elég lesz csak egy modell létrehozása a tanítási folyamathoz? Létre kell-e hozni egy teljesen új modellt, és a nulláról betanítani, hogy megfelelően működjön, vagy elég elvégezni a finomhangolást egy már előre betanított modellen?
7. Állandóan újra kell-e tanítani a modellt, vagy a már betanult modellt el tudom menteni és fel tudom használni termékek kategorizálására?
8. Milyen problémák merülhetnek fel a modell használata közben, és milyen megoldások lehetnek erre?

9. Milyen paraméterekre kell különösen figyelni a finomhangolás során, és hogyan lehet ezeket optimalizálni a maximális pontosság érdekében?

9+1. A legfontosabb kérdés pedig, hogy mennyi ideig fog tartani mindez? Lesz-e egyáltalán rá időm és erőforrásom a saját kis teljesítményű gépemen?

Rengeteg kérdésre kerestem a választ, és maga a projekt is nagyon hosszadalmasnak bizonyult, de véleményem szerint pont emiatt érte meg ezen dolgozni, hogy minél több tapasztalatot tudjak gyűjteni erről a területről.

2. MÓDSZERTAN

A célom, hogy a BERT modellt alkalmazva létre tudjak hozni egy olyan rendszert, amellyel automatizálni lehet egy rengeteg terméket árusító webárúház kategorizálási folyamatát. Ez a megoldás kiváltaná az emberi manuális munkát, illetve javítana a kategorizálás gyorsaságán és pontosságán. A cél elérése érdekében több tényezővel is számolnom kellett, mint például az idővel vagy az erőforrási igényekkel. Egy bonyolultabb mesterséges intelligencián alapuló modell (csak úgy, mint maga a BERT modell is), betanítása időigényes, illetve sok technikai erőforrást igényel, és pont e két tényező szűkössége okozott nehézséget a projektem megvalósítása során.

2.1 A FELADAT DEFINIÁLÁSA

Kezdetben a feladatom az volt, hogy létre kellett hoznom egy olyan BERT modellt, amely termékek kategorizálására specializálódott. Viszont a kutatásaim során rájöttem, hogy nem kell az elejétől betanítanom a modellt, hanem elég csak kiválasztanom egy megfelelőt, aminek az előtanítási folyamata már megtörtént, így elég csak a finomhangolási folyamatot elvégezni, hogy a megfelelő feladatra legyen specializálva, amely ez esetben a kategorizálás volt.

A megfelelő kategorizálás fontossága jelentős egy webáruház számára, különösen egy olyan számára, ahol nagyon sok fajta termék árusításával foglalkoznak. Mindezek mellett az én esetemben a termékek több, mint 2000 kategóriába vannak besorolva. Egy ekkora számnál pedig egy ember nem mindig képes pontosan behatárolni, hogy melyik kategóriába tartozhat egy-egy termék. Épp emiatt fontos egy olyan modell létrehozása, amely pontosan tudja hány darab és milyen fajta kategóriák vannak, illetve azt is ki tudja következtetni, hogy mely termék melyik kategóriába tartozik.

2.2 FEJLESZTÉSI FOLYAMAT

A fejlesztési folyamat során rájöttem, hogy ez a modell sokkal komplexebb, mint amiket az eddigi tanulmányaim során megismertem. A legnehezebb része az volt, hogy meg tudjam érteni a modell logikáját, mi alapján tudja elemezni a szavakat, ami alapján be tud illeszteni

egy teljesen új terméket a megfelelő kategóriába. A teszteléseim során rá kellett jönnöm, hogy a finomhangolási folyamatot több tényező is befolyásolhatja, melyek jelentős hatással tudnak lenni a végeredményre, így mindezekre odafigyelve hoztam létre a végső modellt.

A fejlesztés során főként olyan problémákba ütköztem, hogy a személyesen használt számítógépem nem rendelkezett a megfelelő mértékű erőforrásokkal a finomhangolási folyamat elvégzéséhez és a modell teszteléséhez. Ennek okán ezt egy online környezetben oldottam meg, pontosabban a sokak által használt Google Colab rendszerében. Itt be tudtam állítani, egy nagyobb teljesítményű környezetet is, aminek köszönhetően sikerült viszonylag rövid idő alatt finom hangolnom a modellt, és több tesztelést is végrehajtanom.

2.3 ADATGYŰJTÉS

A felhasznált adatok, amelyek alapján a modellt finomhangoltam primer adatként szolgáltak. Ezeket a webáruház biztosított számomra, egy táblázatos formátumban. Ez az adathalmaz nagyjából 45 ezer terméknévből, illetve azok kategóriájából és alkategóriájából tevődött össze.

A modellről gyűjtött információimat mind hivatalos szaklapokból, tudományos cikkekből és könyvekből gyűjtöttem össze, mint secunder adathalmazokat. Mivel ezek másodkézből származtak, emiatt igyekeztem minél modernebb, de egyúttal hivatalos forrásokat felhasználni, hogy az információim megbízhatóak legyenek. Próbáltam minél változatosabb területekről összeszedni a tudást, hogy ne csak magyar nyelvű forrásokat dolgozzak fel. Viszont mivel az informatika világában a legtöbb tudományos cikk angolul van megírva, inkább a magyar nyelvű cikkek keresése okozott problémát. Nagyon sok esetben találkoztam olyannal is, hogy bár egy adott cég magyar volt, és a témámmal kapcsolatos cikket készítettek el, azt angolul adták ki, ami miatt a forrásaim között inkább angol nyelvű cikkek szerepelnek.

2.4 ADATOK FELDOLGOZÁSA ÉS ELEMZÉSE

Mielőtt magukkal az adatokkal dolgoztam volna, előbb feldolgoztam a másodkézből összegyűjtött irodalmakat. Ezek alapján tanultam meg, hogy hogyan működik maga a modell, illetve milyen erőforrás igényekkel rendelkezik. Ezen cikkek alapján készítettem elő az adathalmazomat is, amelyen az egyetemi tanulmányaim során elsajátított adattisztítási folyamatokat hajtottam végre. Ezek után céltudatosan tudtam nekilátni a modell elkészítéséhez.

Amikor a modell finomhangolásához értem, akkor különös odafigyeléssel kellett dolgoznom az iránt, hogy a tanító adathalmazban a kategória fajtákból mindegyik lehetséges esetből szerepeljen legalább egyszer egy példa. Mivel ennek a modellnek nem az volt a célja,

hogy a betanult adatok alapján létre tudjon hozni új kategóriákat, hanem hogy a termékeket be tudja sorolni a megfelelő kategóriájába, előbb azt biztosítanom kellet, hogy a már meglévő kategóriákat ismerje a modell. A finomhangolási lépéseknél, ahol lehetett, ott szintén alkalmaztam az egyetemen elsajátított tudásomat, ilyen volt például a tanítási folyamat során alkalmazott korai leállás is.

A végeredményeket végül egy táblázatba foglaltam össze, ahol nem találtam jobb opciót, mint hogy a hibás predikciók esetén egyesével ellenőriztem, hogy mi alapján kategorizált rosszul a modell. Az átvilágítás során csoportosítottam az elemeket a szerint, hogy mi lehetett a probléma, majd ezt vizuálisan is ábrázoltam. A végeredmény elemzésénél szintén próbáltam olyan eszközökhöz nyúlni, amiket már ismerek, viszont még nem volt lehetőségem alkalmazni a való életben. Például egyes ábra előkészítéséhez volt, hogy az Excel Power Pivot táblázatkészítője segítségével készítettem elő, de volt, hogy a Tableau alkalmazást használtam, egy olyan táblázat létrehozásához, amely az elvárásaimnak megfelelt.

2.5 MODELL KIVÁLASZTÁSA

A BERT modellnek megjelenése óta rengeteg változata jött létre, és mindegyik másra volt előtanítva. Ennek okán kulcsfontosságú tényező volt egy olyan modellt kiválasztanom, ami a legjobban tudott illeszkedni az adathalmazomra. Mivel a feladatom egy kategorizáló rendszer kialakítása volt, amely képes több kategória változót együttesen kezelni, így elég volt egy kisebb és egyszerűbb modellt választanom. Ennek a modellnek nem kellett bonyolult szövegrészeket feldolgoznia, elég volt csak a nevek és a kategóriák közötti kapcsolatot értelmezni, emiatt nem kellett felépítenem egy teljesen új modellt, hanem kiválasztottam az én célomnak egy megfelelőt, és annak a finomhangolásával foglalkoztam.

Ez volt a legelső mérföldkő, amely meghatározta a projektem kimenetelét. Mivel mindegyik modellnek más volt a betanítási alapja, így mindegyik modellt másra lehetett felhasználni. Példának okán, ha egy olyan modellt választottam volna, amely a jogszabályok alapján tanult elő, az nem biztos, hogy ugyanolyan jó eredménnyel tudta volna felismerni a termékek nevei közötti összefüggést. De ugyan ez lett volna a helyzet, egy olyan modellel is, amely kizárólag angol nyelvű szövegkörnyezetben tanult elő. Ennek hátrányai az eredményekben valószínűleg erőteljesen látszódtak volna, például maga a finomhangolási folyamat is hosszadalmasabb lett volna, mert nehezebben tudta volna értelmezni a szavakat, és azok nyelvtani sajátosságait. Ennek következtében a végeredményben is gyengébb teljesítményt ért volna el a kategorizálás pontosságában.

A sok opciók közül azt a modellt választottam, ami a legalkalmasabbnak tűnt az adathalmazom szempontjából. Ez egy olyan modell volt, ami a magyar nyelvre lett előre betanítva. Az én adataim is erőteljesen támaszkodtak a magyar nyelvre, az az majdnem minden terméknévben volt például ékezet, de olyan is volt, hogy nyelvtani toldalékkal volt ellátva. Ezeket az angol nyelven betanított modellek nem nagyon tudtak volna értelmezni, és emiatt volt kulcsfontosságú tényező egy olyan modell kiválasztása, amely ismeri ezeket a szimbólumokat és szerkezeteket.

3.SZAKIRODALOM

3.1 ALAPVETŐ FOGALMAK ISMERTETÉSE

Amikor mesterséges intelligenciával foglalkozunk, vagy kutatást végzünk a témában, számos fogalommal találkozhatunk, melyek jelentése nem ismert mindenki számára. Ilyen például a gépi tanulás (Machine Learning, ML), mélytanulás (Deep Learning, DL), valamint a természetes nyelvi feldolgozás (Natural Language Processing, NLP). Ezeknek a kifejezéseknek a pontos megértése elengedhetetlen, ha MI területen dolgozunk, hiszen ezek összekeverése könnyen félreértésekhez és szakmai pontatlanságokhoz vezethet.

Emiatt összeszedtem néhány fontosabb fogalmakat és azoknak a definícióit. Az alábbiakban ezeket fogom ismertetni, mivel ezek gyakran előfordultak a kutatási munkám során, és szorosan kapcsolódnak a BERT modellhez.

3.1.1 MI - MESTERSÉGES INTELLIGENCIA

Első lépésként érdemes tisztázni, mit is jelent maga a mesterséges intelligencia. Az MI (angol megfelelője AI – Artificial Intelligence), egy számítástechnikai tudományterület, amely olyan rendszerek és algoritmusok fejlesztésével foglalkozik, amelyek képesek emberi intelligenciát igénylő feladatokat végrehajtani. Története egészen az 1950-es évekig nyúlik vissza, amikor Alan Turing megalkotta a híres Turing-tesztet, ami azt mérte, hogy egy gép képes-e olyan viselkedést mutatni, ami megkülönböztethetetlen az emberi intelligenciától. Ilyen például az érvelés, a mintafelismerés, a tanulás, beszédfelismerés, és a természetes nyelv feldolgozás.

A fejlesztések sajnos időközben elmaradtak akkor, mivel nem volt meg a megfelelő technológiai háttér, viszont a számítógépek megjelenésével megszűnt ez a probléma. Az MI technológiák manapság széles körben alkalmazottak az iparban és a hétköznapi életben, például az automatizált ügyfélszolgálati rendszerekben, személyi asszisztensekben, valamint a gépi látás és a robotika területén. (Russell & Norvig, 2021)

3.1.2 AZ MI-K FAJTÁI

Felhasználási módjuk szerint rengeteg fajtát lehet megkülönböztetni, sőt, van közöttük olyan is, amely jelenleg még nem létezik, csak elméleti síkon. Jelen kutatásom során azokat

ismertetem, amelyek a legszorosabban állnak a témához. Ilyen például a szűk MI (narrow AI), általános MI (general AI) vagy éppen a Generatív MI (Artificial Generative Intelligence - AGI). Az MI kategóriák közötti különbség alapvetően abban rejlik, hogy milyen típusú feladatokat képesek elvégezni és milyen módszereket alkalmaznak.

A szűk MI egy adott feladatra specializált mesterséges intelligencia. Képes egy konkrét problémát megoldani, de nem rendelkezik általános intelligenciával, így nem tud más, egyéb területeken működni. Ezek a rendszerek olyan speciális feladatok elvégzésére készültek, mint például arcfelismerés, nyelvi fordítás, spam-szűrés vagy játékokban való részvétel.

A generatív mesterséges intelligencia olyan MI-rendszer, amely képes új, eredeti tartalmak előállítására (például szövegek, képek, zene), amihez a tanulás során szerzett mintákból merít inspirációt. Az ilyen rendszerek, mint például a GPT (Generative Pre-trained Transformer) a meglévő adatok alapján képesek új tartalmakat generálni, mint például szövegek írása, képek vagy zenék létrehozása. A generatív mesterséges intelligencia működésének alapját a generatív modellek képezik, amelyek valósághű minták vagy adathalmazok előállítására képesek. Ezek a modellek gépi tanulási algoritmusokon alapulnak, amelyek a tanulási folyamat során elsajátított adatokból hoznak létre új mintákat. A generatív modellek az adatok eloszlását modellezik, majd olyan új adatokat állítanak elő, amelyek hasonlóak a tanult mintákhoz, de mégis egyediek, és nem másolják közvetlenül az eredeti adathalmazt. (Goodfellow, et al.,2016)

Az általános MI pedig egy olyan elméleti koncepció, amely célja, hogy az emberi intelligenciához hasonló rendszert hozzanak létre. Ez a típusú mesterséges intelligencia nem csak specifikus feladatokat oldana meg, mint a generatív MI, hanem bármilyen új problémával szembesülve képes lenne tanulni és alkalmazkodni. Az általános MI rendelkezne olyan képességekkel, mint a következtetés, a döntéshozatal, a kreatív gondolkodás és a tanulás. Például egy ilyen modell képes lenne nem csak egy sakkozó programot működtetni, hanem megtanulni a sakkstratégiákat, értelmezni az ellenfél gondolkodását, és akár új stratégiákat kidolgozni. Viszont ilyen szintű intelligenciát az emberiség még nem tudott kifejleszteni, jelenleg nagyon kevés ismerettel rendelkezünk az agy működéséről, ami még nem elég, ahhoz, hogy egy ilyen rendszert képesek legyünk létrehozni. A fejlesztéséhez rengeteg területről kellene tudást összegyűjteni, mint például a számítástechnika vagy a pszichológia. Mindezek okán ez a fajta MI inkább tudományos elképzelésként vagy célként létezik jelenleg a mesterséges intelligencia kutatásban. (Innodata, 2023)

3.1.3 A GÉPI TANULÁS (MACHINE LEARNING)

A gépi tanulás az MI egyik ága, amely olyan algoritmusok fejlesztésével foglalkozik, amelyek képesek tanulni és javulni az adatok alapján anélkül, hogy explicit[1] módon programoznák őket minden egyes feladatra. A gépi tanulás során a rendszer az adatokból mintákat és összefüggéseket fedez fel, majd ezek alapján predikciókat készít, illetve döntéseket hoz. A cél az, hogy az algoritmusok folyamatosan jobb eredményeket produkáljanak a bejövő információk alapján, anélkül, hogy minden lépésüket előre meghatároznánk. (IBM, 2023)

A gépi tanulás főbb típusai:

- Felügyelt tanulás,
- Részben felügyelt tanulás,
- Nem felügyelt tanulás,
- Megerősítéses tanulás.

A felügyelt tanulás olyan módszer, amely során az algoritmus címkézett adatokkal dolgozik, azaz a tanulási folyamatban minden adatponthoz tartozik egy célérték vagy címke. Ezeket az adatokat sokszor függő, kimeneti vagy célváltozóknak is szoktuk elnevezni. A bemeneti változókat pedig prediktoroknak, független vagy magyarázó változóknak hívjuk. Az algoritmus célja, hogy a függő és független változók közötti összefüggéseket megtanulja. (Kovács Endre, 2024). Például arcfelismerő rendszerek esetén a képeket címkézett példákkal látják el, ahol a címkék az egyes személyek azonosítói.

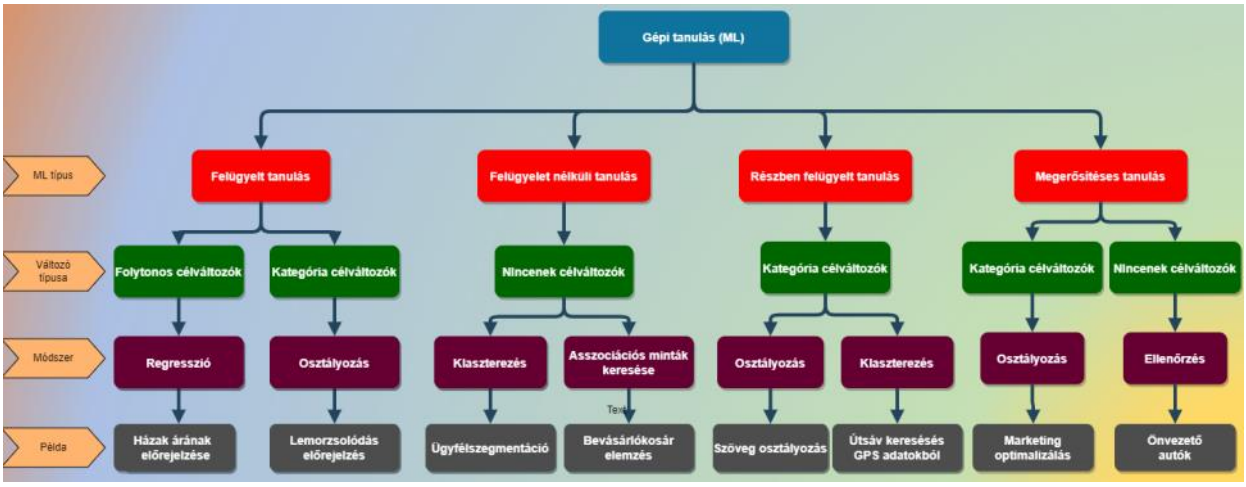
A nem felügyelt tanulás címkézetlen adatokkal dolgozik, azaz csak bemenet van, kimeneti változó nincs. Ilyen esetben az algoritmusnak önállóan kell felfedeznie a rejtett struktúrákat az adathalmazban. Ezeket a módszereket alkalmazzák például klaszterezésre vagy asszociációs szabályok keresésére. Klaszterezési feladat esetén az algoritmus megpróbálja az adatokat természetes csoportokba rendezni, amelyeket előzetesen nem határoztak meg (Aggarwal, 2015). Ilyen megoldásokat használnak például ügyfél-szegmentálásnál marketingben, ahol különböző vásárlói csoportokat azonosítanak a viselkedési minták alapján.

A részben felügyelt tanulás esetén az adataink egy része címkézett, míg egy másik része nem. Ez a módszer a felügyelt tanulás (ahol minden adat címkézett) és a felügyelet nélküli tanulás (ahol nincsenek címkék) között helyezkedik el. A részben felügyelt tanulás hasznos, amikor csak korlátozott mennyiségű címkézett adat áll rendelkezésre, de a címkézetlen adatok könnyen hozzáférhetők. Így a rendszer képes lesz a kis mennyiségű címkézett adatokból

[1] Explicit utasítások: Olyan világosan és részletesen megfogalmazott utasítások, amelyek pontosan meghatározzák, mit kell tenni egy adott feladat elvégzéséhez.

általánosítani, miközben a címkézetlen adatokat is bevonja a tanulási folyamatba, ezáltal pontosabb eredményeket produkál. (Sarmiento, 2024)

A megerősítéses tanulás során az algoritmus interakcióba lép a környezetével, és visszajelzések alapján tanul, ahol jutalmakat vagy büntetéseket kap az egyes döntéseiért. A cél, hogy az algoritmus megtanulja, mely lépések vezetnek a hosszú távon legnagyobb jutalomhoz. Ezt a módszert gyakran használják autonóm rendszerek, például robotok és önvezető járművek esetében, valamint stratégiai játékokban (Sutton & Barto, 2018).



1. ábra: A gépi tanulás típusai

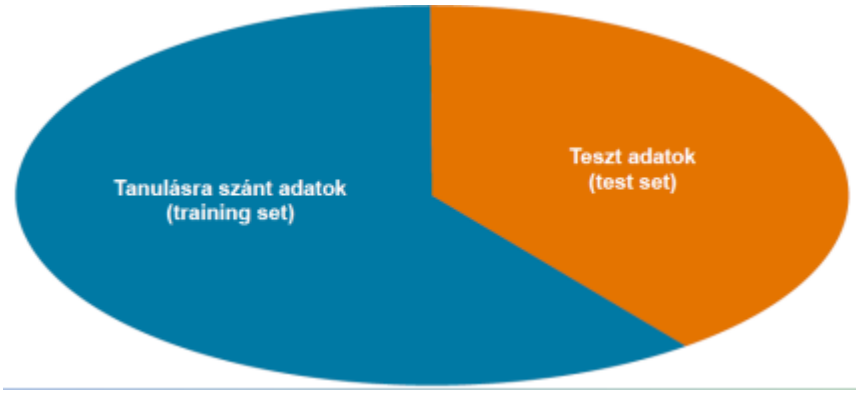
Forrás: Kovács Endre, Mesterséges Intelligencia előadás, 2024

Az 1. ábra bemutatja a gépi tanulás típusait, melyekhez a jellemző alkalmazási módszereket is hozzárendeli. Továbbá azt is szemlélteti, hogy a különböző tanulási típusok hogyan alkalmazhatók a valós világ különböző problémáinak megoldására. A különböző tanulási módszerek lehetőséget adnak arra, hogy a gépi tanulási algoritmusokat az adott probléma természetéhez igazítsuk, így javítva a teljesítményt és a hatékonyságot.

A gépi tanulást nagyon sok területen alkalmazzák. Például az arcfelismerés során a rendszerek olyan képelemzési algoritmusokat alkalmaznak, amelyek az emberi arcok jellemzőit azonosítják és osztályozzák. Az ajánlórendszerek, melyeknek célja, hogy személyre szabott javaslatokat tegyenek a felhasználóknak a korábbi interakcióik alapján, mint például filmek vagy termékek ajánlása online platformokon, szintén gépi tanulást használnak. A hangfelismerés pedig arra szolgál, hogy emberi beszédet gépi utasításokká alakítson át, amelyeket az algoritmusok képesek értelmezni.

A mesterséges intelligencia és a gépi tanulás közötti különbség az, hogy míg az előbbiben a kutatásaiban a cél orientált megközelítések dominálnak, azaz a kutatás tárgya az intelligens viselkedést létrehozó algoritmusok és matematikai összefüggések feltárása, addig a

gépi tanulási algoritmusok lényege, hogy statisztikai módszerekkel modellezik az adatok közötti kapcsolatok mintázatait, gyakran regressziófüggvények segítségével. A tanulási folyamat alapját a tanuló- és teszt adathalmazok képezik, amelyeket a változók közötti kapcsolatok megértésére használnak (Hastie, et al., 2009).



2. ábra: Tanító- és teszt adathalmazok aránya

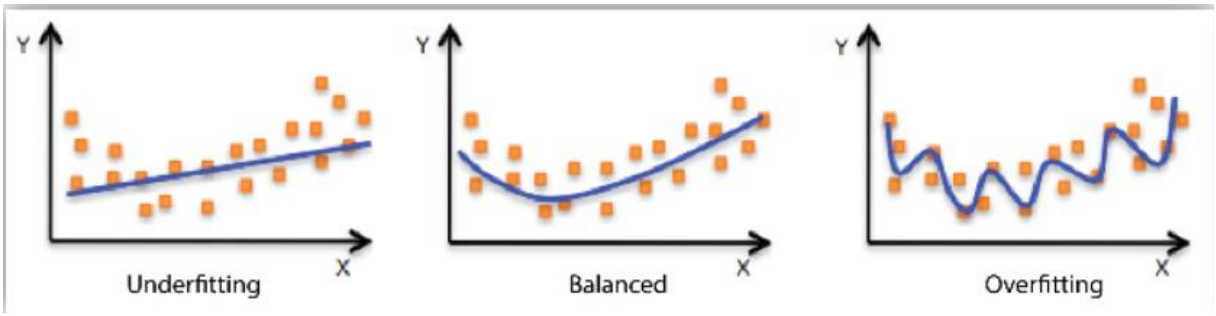
Forrás: Kovács Endre, Mesterséges Intelligencia előadás, 2024

Tanító adatoknak nevezzük az adathalmaz azon részét amely a modell betanítását szolgálja. Ezek alapján történik meg az adott modell finomhangolása, amelynek köszönhetően tud a későbbiekben prediktálni. A teszt adatok pedig a modell validásában segítenek, tehát ezeket használjuk fel arra, hogy leteszteljük, hogy a modell mellyire jól tud előrejelezni, mennyire túltanult, vagy épp alultanult. A 2. ábra mutatja, hogy általában milyen arányokkal szokták szétválasztani a tanító és tesztadatokra az adathalmazt, ha egy gépi modellt szeretnénk betanítani. A szakirodalom azt írja, hogy általában 60-80%- ot szoktak tanító adatként használni. Mikor én választottam szét az adataimat, akkor ennek az útmutatásnak megfelelően 80%-os aránnyal dolgoztam.

A modellek teljesítményének értékelése költségfüggvények segítségével történik, amelyek azt mérik, hogy az előrejelzések mennyire térnek el a valós kimenetektől. Az ideális modell alacsony torzítással rendelkezik (azaz pontosan leírja a valós világban fennálló kapcsolatokat), és alacsony változékonysággal bír (azaz különböző adatmintákon stabil teljesítményt nyújt).

A modell összetettsége jelentősen befolyásolja annak teljesítményét. Ha egy modell túl egyszerű, magas torzítással rendelkezik, és nem képes megfelelően leképezni az adathalmaz bonyolult kapcsolatait. Ezt a jelenséget alultanulásnak nevezik, és ilyenkor az algoritmus nem képes hatékonyan megérteni az adatban lévő mintázatokat. A másik véglet a túltanulás, amikor is a modell túlságosan jól illeszkedik a tanító adathalmazra, azonban az új, nem látott adatokon nem teljesít jól. A túltanulás során a modell "megtanulja" a zajt is az adathalmazban,

így túlzottan rugalmas lesz, és nem tud jól általánosítani (Domingos, 2012). A 3. ábra szemlélteti, hogy melyik esetben, hogy szokott kinézni a tanulási folyamat.



3. ábra: Túltanulás és alultanulás

Forrás: Amazon Machine Learning, 2024

Az ideális modell egyensúlyt teremt az alul- és túltanulás között: elegendően összetett ahhoz, hogy az adatokban rejlő valódi mintázatokat felismerje, de nem annyira bonyolult, hogy túlzottan illeszkedjen a tanító adathalmazra. A modellek ezen képességét, hogy új, eddig nem látott adatokon is jól teljesítsenek, általánosításnak nevezzük. Az általánosító képesség rendkívül fontos, különösen olyan alkalmazásoknál, ahol a modellnek gyorsan változó körülmények között kell döntéseket hoznia, mint például a pénzügyi piaci elemzésekben vagy az önvezető autóknál.

3.1.4 MÉLYTANULÁS (DEEP LEARNING)

A mélytanulás a gépi tanulás egy speciális területe, amely mesterséges neurális hálózatok segítségével dolgozza fel az adatokat, és képes bonyolult mintázatok felismerésére. A módszer alapja a több rétegből álló neurális hálózatok használata, ahol minden réteg egyre magasabb szinten értelmezi a bemeneti adatokat. Ez a hierarchikus feldolgozás hasonló az emberi agy működéséhez, amelyben az információ fokozatosan, különböző szinteken kerül feldolgozásra.

A mélytanulás egyik nagy előnye, hogy automatikusan képes felfedezni az adatokban található fontos jellemzőket, anélkül, hogy manuális beavatkozásra lenne szükség. Ez különösen hasznos nagy méretű adathalmazok esetén, és olyan összetett feladatoknál alkalmazzák sikeresen, mint a képfelismerés, a természetes nyelvfeldolgozás és a beszédfelismerés. A rétegek mélysége lehetővé teszi, hogy a modellek felfedezzék az adatokban rejlő komplex struktúrákat, ami különösen hasznos a nagy és változatos adatok kezelésében. (Gyires-Tóth, 2020)

3.1.5 TERMÉSZETES NYELVI FELDOLGOZÁS (NLP)

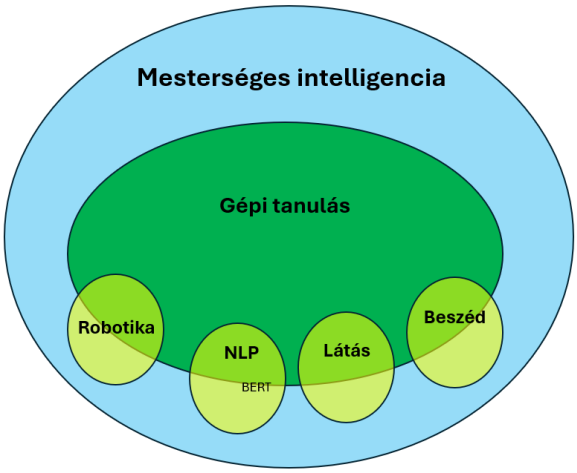
A természetes nyelvi feldolgozás (Natural Language Processing, NLP) az MI egyik ága, amely a számítógépek és az emberi nyelvek közötti interakcióval foglalkozik. Az NLP célja,

hogy a gépek képesek legyenek megérteni, értelmezni és generálni az emberi nyelvet oly módon, hogy az természetes és hasznos legyen a felhasználók számára.

Az NLP feladatai közé tartozik a szövegek elemzése, a nyelvtani szerkezetek felismerése, a szavak és kifejezések jelentésének megértése, valamint a szövegből való információ kinyerése. Ezen kívül az NLP alkalmazható fordítási rendszerekben, beszédfelismerésben, csevegőbotokban, valamint olyan eszközökben, amelyek automatikusan képesek szöveget generálni vagy összefoglalni. A természetes nyelvi feldolgozásban használt gépi tanulási modellek, például a BERT vagy a GPT, képesek arra, hogy az emberi nyelvet mélyen megértsék és bonyolult nyelvi feladatokat oldjanak meg. Az NLP különösen fontos, mert lehetővé teszi, hogy a gépek természetes módon kommunikáljanak az emberekkel, és különféle nyelvi problémákra találjanak hatékony megoldásokat, mint például a gépi fordítás vagy az ügyfélszolgálati automatizálás. (Nelson, 2024)

3.2 A MODELL ELHELYEZKEDÉSE A MESTERSÉGES INTELLIGENCIA HALMAZÁBAN

A fogalmak közötti kapcsolat megértéséhez, illetve, hogy közöttük hol helyezkedik el a BERT, amelyet a 4. ábra mutat:



4. ábra: BERT modell elhelyezkedése

Forrás: Kovács Endre, Mesterséges Intelligencia előadás, alapján kismértékben módosítva 2024

Jól látható, hogy a legnagyobb kör az MI-t (mesterséges intelligenciát) reprezentálja, amely egy átfogó kifejezés az emberi intelligenciát utánzó gépekre és algoritmusokra. Az MI-n belül található a gépi tanulás (ML), amely az MI egy része, és egy olyan módszertani megközelítés, amely lehetővé teszi a rendszerek számára, hogy nagy mennyiségű adatokból tanuljanak, azaz mintázatokat ismerjenek fel és előre nem beprogramozott döntéseket hozzanak ezek alapján.

Ez a terület különösen fontos az MI számára, mivel lehetővé teszi, hogy a rendszerek folyamatosan javítsák teljesítményüket és alkalmazkodjanak az új információkhoz.

A 4. ábrán, ami a témával kapcsolatban a legfontosabb még az NLP. Jól látható, hogy az NLP nem teljesen tartozik a gépi tanulás alá, ennek oka, hogy nem kizárólag gépi tanulásra épül, hanem magában foglal olyan technikákat és megközelítéseket is, amelyek más módszereken alapulnak.

3.3 A KATEGORIZÁLÁS JELENTŐSÉGE NAGYVÁLLALATI SZEMPONTBÓL

Egy olyan nagyvállalat sikeres működése szempontjából, mint amilyennel én is dolgoztam, elengedhetetlen a hatékony kategorizálási módszer megválasztása, amely a vásárlói élmény és a keresőoptimalizálás (SEO - Search Engine Optimization, magyarul kereső optimalizálás) szempontjából is meghatározó szerepet játszik. A jól kialakított kategóriaszerkezet leegyszerűsíti a böngészést, csökkenti a keresési időt, és növeli a látogatók elégedettségét. Egy összetett, átgondolatlan kategóriaszerkezet könnyen zavart kelthet a vásárlókban, és akár az oldal elhagyásához is vezethet, hiszen, ha a látogató nem találja könnyen a kívánt terméket, inkább más oldalt keres, amely gördülékenyebb vásárlói élményt biztosít.

Például, ha egy vevő meg szeretne venni egy adott terméket a webáruházban, viszont nem tudja pontosan, hogy milyen típusúak vannak készleten, akkor megnézi az adott kategórián belül a termékeket. Ilyenkor alapvetően feltételezi, hogy csak annyi elérhető, amennyit ott lát, és nem kezd el tovább böngészni a többi kategória között. Így még ha van is készleten az adott termékből, ha a vevő nem találja meg azonnal, amit keres, akkor inkább tovább áll, és keres egy másik webáruházat, hátha ott könnyebben rátalál. Ekkor a cég potenciális vásárlástól eshet el, ami pedig bevétel kiesést okoz.

A megfelelő kategorizálás például különösen fontos a műszaki termékeknél, ahol számos terméktípus és szűrési lehetőség szokott előfordulni. Emellett a megfelelően kialakított kategóriastruktúra fontos a keresőmotorok számára is, mivel a kategórianevek optimalizálása növeli a találati listán való helyezést. Ezáltal a webáruház több potenciális látogatót vonz, akik nagyobb valószínűséggel válnak vásárlókká. (Gyetvai, 2023)

A kategórianevek optimalizálásában a SEO rendszer segít. Olyan technikákat és módszereket foglal magába, amelyek segítségével egy weboldal jobb helyezést érhet el a keresőmotorok találati listáján. A SEO célja, hogy a weboldal minél relevánsabbnak tűnjön a keresőmotorok számára, ami növeli a látogatottságot és segíti a potenciális vásárlókat vagy olvasókat abban, hogy könnyebben megtalálják a tartalmat.

A kategóriastruktúra kialakításánál érdemes figyelembe venni, hogy túl sok vagy túl kevés kategória használata egyaránt problémákat okozhat. Ha túl sokat hozunk létre, az

zavaróvá válhat, és elriaszthatja a vásárlókat. Ezzel szemben, ha túl kevés kategória van, a látogatók nehezen találják meg a keresett termékeket. Az optimalizált struktúra kialakítása érdekében érdemes a kategóriákat rendszeresen tesztelni, és vásárlói visszajelzések alapján finomítani. (Nagy, 2024)

4 A BERT MODELL

4.1 A BERT MODELL FOGALMA

A BERT egy 2018-ban a Google által kifejlesztett nyelvi modell, amely forradalmasította a természetes nyelvfeldolgozás területét. A BERT legfőbb újítása, hogy a szöveg értelmezéséhez kétirányú kontextust használ, vagyis egyszerre veszi figyelembe a szöveg balról jobbra és jobbról balra történő olvasatát. Ez különösen hasznos az emberi nyelv értelmezésénél, ahol egy szó jelentése sokszor a környező szavak függvényében változik

Ennek a módszernek a segítségével a modell különösen hatékonynak bizonyult számos nyelvi feladatban, mint például a kérdés-válasz rendszerekben, szövegértelmezésben és mondatosztályozásban is. A modell további előnye, hogy egyetlen előképzett modellt is több feladatra lehet felhasználni, példának okán, amit én használtam modell eredetileg magyar szövegek értelmezésére lett betanítva, viszont én a finomhangolás során terméknevek kategorizálására használtam. Sok cég használja ezt a modellt kategorizálás szempontjából is, csak úgy, mint például az Amazon. (Koroteev, 2021)

Viszont hátrányai közé lehet sorolni, hogy rendkívül nagy a számítási kapacitás igénye, pont a bidirekcionális kontextus feldolgozása miatt. Egy teljes BERT modell betanítása és finomhangolása jelentős hardverkapacitást követel, ami megnehezíti a használatát kisebb eszközökön vagy valós idejű alkalmazásokban. Az is negatívum benne, hogy bár jól teljesít a kontextusértelmezési és alapvető nyelvi feladatokban, a mélyebb logikai következtetéseknél csak korlátozott eredmények elérésére képes. Például nem tud jól kezelni olyan feladatokat, amelyek mélyebb ok-okozati mégértést igényelnek. (Rogers, et al., 2021)

A BERT modell adta előnyöket mi magunk is megtapasztalhattuk, amikor a Google 2019-ben beépítette a keresési algoritmusába. A modell bevezetése jelentős javulást hozott a Google keresési algoritmusában, különösen a keresési lekérdezések szövegének értelmezésében. Korábban a Google főként a kulcsszavak azonosítására és ezek alapján történő találat-rendezésre koncentrált, de a BERT használatával képes lett a keresési lekérdezések teljes kontextusát figyelembe venni. Ez különösen hasznos olyan hosszabb, bonyolultabb kérdések esetében, ahol a keresés szándéka nem közvetlenül a kulcsszavak alapján érthető meg. Például a „utazás New Yorkból Los Angelesbe” és az „utazás Los Angelesből New Yorkba” lekérdezések közötti különbséget a BERT segíti felismerni,

figyelembe véve az irányokat és a prepozíciókat, amelyek korábban félreértelmezhetők voltak.(Alwis, 2020.)

4.2 TÖRTÉNETI HÁTTERE ÉS HATÁSA A JELENRE

A BERT modell történelmi háttere szorosan összefügg a mélytanulás és a természetes nyelvfeldolgozás fejlődésével, valamint a transformer architektúrával. Ezen architektúra megjelenése előtti időkben a természetes nyelvfeldolgozás területén az uralkodó megközelítések gyakran sorrendi (szekvenciális) modellek voltak, mint például a rekurzív neurális hálózatok (RNN) és a hosszú távú memóriával rendelkező hálózatok (LSTM). Ezek a modellek főként a mondatok szavainak egyirányú feldolgozására voltak alkalmasak, vagy balról jobbra, vagy jobbról balra olvasták a szöveget. Ez korlátozta azt a képességüket, hogy teljes mértékben kihasználják a szöveg mindkét oldalának kontextusát, ami jelentős hátrány volt a pontos nyelvfeldolgozás szempontjából.

A transformer architektúrát 2017-ben mutatták be az *Attention is All You Need* című tanulmányban. Ennek a lényege, hogy a szövegek feldolgozásához önfigyelési mechanizmust használ, amely a módszer lehetővé tette, hogy mindegyik szó figyelembe vegye a többi szóval való kapcsolatát a szövegen belül. Ezzel megszűnt a hagyományos szekvenciális feldolgozás szükségessége, és jelentős javulást eredményezett a sebesség, illetve a hatékonyság terén. A BERT modell is ezt az architektúrát alapul véve működik, és ennek köszönhetően képes kétirányú megfigyelést alkalmazni a bemeneti adatra, így egyszerre képes figyelembe venni a szöveg balról jobbra és jobbról balra értelmezhető kontextusát is. Ennek eredményeképpen a modell jobban megérti a szavak és mondatok közötti összefüggéseket, hiszen a szövegkörnyezetet teljes egészében vizsgálja. Fontos azonban kiemelni, hogy a szavak sorrendjét továbbra is felhasználja a modell. (Devlin, et al., 2019)

A BERT számos nyelvi feldolgozási referenciafeladaton, például a General Language Understanding Evaluation (GLUE) és a Stanford Question Answering Dataset (SQuAD) teszteken kiemelkedő eredményeket ért el. Ezek a tesztek a nyelvi megértés különböző aspektusait mérik, és a BERT sikerei ezen a területen hozzájárultak ahhoz, hogy az egyik legfontosabb kutatási és fejlesztési modellé váljon az NLP-ben.

A BERT megjelenése után számos továbbfejlesztett verzió született. Ezek közé tartozik például a RoBERTa (Robustly optimized BERT approach), amely a BERT tréning folyamatát finomította, és általánosságban még jobb eredményeket ért el. További variánsok, mint például az ALBERT (A Lite BERT) és a DistilBERT, pedig a modell hatékonyságának növelését és méretének csökkentését célozták.

4.3 A MODELL ALAPVETŐ MŰKÖDÉSE

A modell történelmi hátterének áttekintése során megismerkedtünk azzal, hogy ez egy transformer architektúrára épülő nyelvi modell, amely mély tanulási technikákat alkalmaz a természetes nyelv megértéséhez. Az architektúra két fő részre oszlik, kódolóra és dekódolóra. A kódoló feladata a bemeneti adatok feldolgozása, és a bennük található kapcsolatok felfedezése és rögzítése. A szöveg értelmezéséhez önfigyelő mechanizmusokat, illetve előre csatolt neurális hálózatokat is alkalmaz. Kimenete pedig egy nagy dimenziós vektorsorozat, amely nemcsak az egyes elemeket írja le, hanem azt is, hogy ezek hogyan kapcsolódnak egymáshoz a teljes szövegben. Ezt a vektorsorozatot használja fel a dekódoló, melynek célja további kimeneti adatok generálása a kódoló által biztosított bemenet és az eddig generált kimenet alapján. A dekódoló rétegeiben szintén találhatók önfigyelő mechanizmusok, de ezek maszkoltak, hogy biztosítsák az adatok sorrendben történő generálását. A maszkolás lényegében azt jelenti, hogy a dekódoló korlátozza, mely adatokat használhatja a feldolgozás során, így biztosítva, hogy egy adott pozícióban csak a már elkészült korábbi adatokat lássa, így megőrzve a sorrendiséget a kimeneti adatok létrehozásakor. (Aayush, 2023)

A BERT a kódoló részét használja a transformer architektúrának. A modell több rétegű, és minden réteg mélyebb reprezentációt nyújt a szöveg egyre összetettebb összefüggéseiről. Két fő változata van, amelyeket még két-két ágra lehet szétosztani, felhasználásuktól függően:

- BERT-Base kis- és nagybetűket megkülönböztető,
- BERT-Base kis- és nagybetűket nem megkülönböztető,
- BERT-Large kis- és nagybetűket megkülönböztető,
- BERT-Large kis- és nagybetűket nem megkülönböztető.

A BERT-Base és a Large közötti különbségeket pedig az 1. táblázat mutatja be:

1. táblázat: Bert-Base és Bert-Large közötti különbség

	BERT-Base	**BERT-Large**
Kódoló rétegek száma	12	24
Rejtett dimenziói	768	1024
Figyelmi mechanizmusok	12	16
Paraméter	110M	340M
Processzor	4TPUs	16TPUs
A tréning hosszúsága	Kevesebb ideig tart	Több ideig tart

Forrás: (Vaswani et al., 2017)

A BERT-Base egy kisebb modell, amely 12 rétegből áll, és a belső rejtett dimenziója 768. Ezzel szemben a BERT-Large modell sokkal nagyobb, 24 darab kódoló réteggel rendelkezik, illetve a dimenziója 1024. A figyelmi mechanizmusok terén is különbségek vannak: a BERT-Base 12 figyelmi fejet használ, míg a BERT-Large esetében 16 figyelmi mechanizmus van jelen, ami lehetővé teszi a modell számára, hogy több szempontot tartson szem előtt az adatok feldolgozása során.

A paraméterek száma a BERT-Base esetében 110 millió, míg a BERT-Large modellnél ez a szám jelentősen magasabb, 340 millió. Ez azt mutatja, hogy a BERT-Large sokkal nagyobb kapacitással rendelkezik, ami lehetővé teszi számára, hogy bonyolultabb összefüggéseket és mélyebb reprezentációkat tanuljon meg. Azonban ez a nagyobb kapacitás több erőforrást is igényel: míg a BERT-Base modell 4 TPU egységet használ a tanításhoz, addig a BERT-Large 16 TPU-t igényel.

Ez a különbségtáblázat rávilágít arra, hogy míg a BERT-Base egy gyorsabb és kisebb erőforrással működő modell, a BERT-Large több adatot képes feldolgozni és mélyebb megértést kínál, ám mindezt jelentősen nagyobb számítási költségért.

5. A MODELL ARCHITEKTÚRÁJA

Ahogy korábban említettem, a BERT modell a transformer architektúrára épül. A transformer modell két fő részből áll: az encoder és a decoder blokkból, és ezek közül az egyes modellek eltérően használják ezeket. Nagyon jó példa erre a két blokkra a BERT és a GPT (Generative Pre-trained Transformer). Az előbbi modell kifejezetten az encoder részt használja, amely alapján képes a szöveg kétirányú kontextuális értelmezésére, amelyet már fentebb részleteztem. A GPT viszont egy decoder-only modell, amely alapvetően generatív feladatokra lett optimalizálva, mint a szövegalkotás vagy kiegészítés.

Viszont mielőtt ezt a két modellt részletesebben is összehasonlítanám, előbb magáról a transformer architektúráról tartok egy részletesebb áttekintést. Ezáltal egy átfogóbb képet kapunk a működéséről, és jobban meg tudjuk érteni a szerepét. Részletesen leírom a komponenseit, hogy melyik mit csinál, és hogy miért lényeges az architektúra számára.

5.1 A TRANSFORMER ARCHITEKTÚRA

A transformer alapja a figyelem mechanizmus, amely lehetővé teszi, hogy a modell a bemeneti adatok minden elemére fókuszáljon anélkül, hogy időbeli sorrendiséget követne.

Főbb komponensei:

- **Ön-figyelési mechanizmus** (Self-Attention Mechanism): A transformer architektúrában a self-attention mechanizmus biztosítja, hogy a bemeneti sorozat

egyes elemei kapcsolatba lépjenek más elemekkel a sorozaton belül, figyelembe véve azok fontosságát. Ez lehetővé teszi a globális összefüggések kezelését a bemenetek között.

- **Többrétegű encoder-decoder felépítés:** Az architektúra két fő részből áll: egy encoder (kódoló) és egy decoder (dekódoló) részből. Az encoder feladata a bemeneti adatok reprezentációjának előállítása, míg a decoder ezen reprezentáció alapján generálja a kimenetet (pl. egy nyelvi modelltanítás során szöveget).
- **Pozíciós kódolás** (Positional Encoding): A transformer nem tartalmaz rekurrens[2] elemeket, tehát nincs természetes időbeli sorrendiség a bemenetekben. Ennek kezelésére a modellbe pozíciós kódolást építenek, amely hozzáadja a szekvencia időbeli információit a bemeneti vektorokhoz.
- **Teljesen kapcsolt hálózat** (Feed-forward layer): Olyan neurális hálózati komponens, amelyben minden bemeneti neuron összeköttetésben áll minden kimeneti neuronnal. Ez a hálózat a kódoló és a dekódoló minden rétegében megtalálható. Célja, hogy a bemeneti vektorokat lineáris transzformáción és nemlineáris aktivációs függvényen keresztül alakítsa át, így biztosítva, hogy a bemeneti adatok reprezentációja tovább finomodik. A teljesen kapcsolt réteg minden pozícióra külön-külön van alkalmazva, tehát nincs kölcsönhatás a különböző szekvenciális pozíciók között.
- **Rétegnormálás (Layer Normalization):** Ez egy olyan technika, amely stabilizálja a hálózat tanulási folyamatát azáltal, hogy minden réteg bemeneti adatainak normalizálását végzi. Ez azt jelenti, hogy a bemenetek átlagát és szórását szabványosítja, így minden réteget egy egységes skálán dolgoztathat. Ez a normalizáció segít abban, hogy a hálózat gyorsabban és stabilabban tanuljon, mivel csökkenti az adatok extrém ingadozásait, amelyek lassíthatnák vagy megzavarhatnák a tanulást. Ez különösen fontos a mély hálózatok esetében, mint a transformer.
- **Maradék kapcsolat** (Residual Connection): A maradék kapcsolatok, más néven "residual connection" vagy "skip connection", azt jelentik, hogy a hálózat egy rétegének kimenetéhez hozzáadjuk az eredeti bemenetet. Így az aktuális réteg által végzett transzformáció mellett az eredeti bemenet is tovább adódik a következő rétegnek. Ez megkönnyíti a tanulást, mert megakadályozza, hogy az információ

[2] Rekurrens: olyan komponens vagy mechanizmus, amely az előző lépések információit használja fel a következő lépések feldolgozásában.

eltűnjön a mélyebb rétegekben, és javítja a gradiens[3] visszaterjedését, ezáltal hatékonyabbá téve a tanulási folyamatot.

A transformer az inputokat először a self-attention mechanizmus segítségével dolgozza fel, hogy figyelembe vegye a szavak közötti kapcsolódási mintákat. Ezután a feldolgozott adatokat továbbadja az kódoló hálózatnak, ahol több rétegen keresztül zajlik az információk összesítése és elemzése. A kimenet generálásához a dekódoló hasonló figyelem mechanizmusokat használ.

Mivel a transformer nem dolgozza fel sorosan az adatokat, jelentős előnyt nyújt a párhuzamosítás szempontjából, ami gyorsabb számítási sebességet eredményez. A self-attention pedig lehetővé teszi, hogy a modell a bemenet minden egyes elemét összevesse a többi elemmel, ami különösen előnyös hosszú távú függőségek kezelésekor.

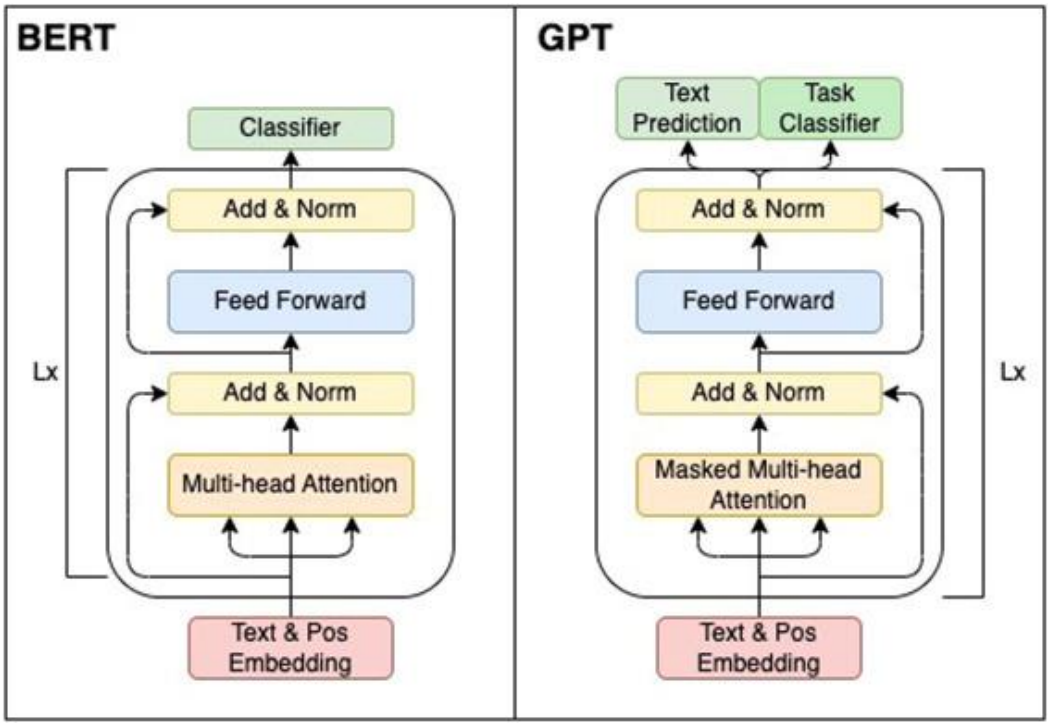
Ez a módszer az imént bemutatott előnyei miatt nagyon elterjedt, és sok más modell is alkalmazza, mint például az egyik leghíresebb MI modell a GPT és változatai is.

5.2 A BERT ÉS GPT KÖZTI KÜLÖNBSÉG

A transformer architektúra két meghatározó nyelvi modellje, a BERT és a GPT, eltérő módon hasznosítja a rendszert. Ezt az alábbiakban fogom részletezni, kiemelve, hogy mi a két modell között a különbség.

A GPT modell autoregresszív dekódolást használ, ami azt jelenti, hogy a bemenetként szolgáló szöveg alapján generálja a következő szavakat, kizárólag a bal oldali kontextusra támaszkodva. Ez az eljárás lehetővé teszi a GPT számára, hogy folyamatosan, szóról szóra építse fel a szöveget, kizárva a jobbról jövő információt, ami ideális a szöveggenerálási feladatokhoz. Ezzel szemben a BERT a transformer kódoló részét alkalmazza, amely bidirekcionálisan, azaz egyszerre veszi figyelembe mind a bal, mind a jobb oldali kontextust, ami nagyobb pontosságot biztosít a szövegfeldolgozás során.

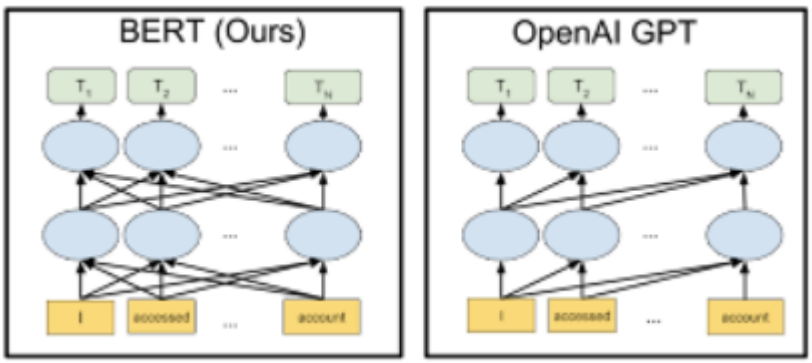
[3] Gradiens: a tanulási folyamat során a hálózati súlyok változtatásának irányát és mértékét határozza meg.



5. ábra: A BERT és a GPT összehasonlítása

Forrás: Qiu & Jin, 2024

Ahogyan az 5. ábra is mutatja, mindkét modell a bemeneti szövegekhez pozíciós beágyazást (Text & Pos Embedding) alkalmaz, amely biztosítja, hogy a szavak sorrendjét a rendszer figyelembe vegye. A BERT és a GPT is többfejű figyelmi mechanizmust (Multi-head Attention) használ, azonban a BERT-ben ez teljes mértékben bidirekcionális, míg a GPT-ben maszkolt figyelmet alkalmaznak, amely kizárólag a bal oldali kontextusra összpontosít, korlátozva a modell által látott információkat a korábbi szavakkal kapcsolatban. Ennek a szakasznak a két modell folyamatában lévő különbségét a 6. ábra mutatja be.



6. ábra: a BERT és a GPT maszkolási eljárásai

Forrás: Devlin & Chang, 2018

A BERT modell is használ maszkolási eljárást, viszont az különbözik a GPT által használttól, ahogyan a 6. ábra is mutatja. A BERT modell által használt maszkolási eljárást majd később, a 6.1 Pre-training folyamat fejezetben részletezem.

Mindkét modell rétegeiben normalizálást és összegzést (Add & Norm) végeznek, ami stabilizálja a tanulást és csökkenti a túlillesztés kockázatát. A BERT-ben és a GPT-ben egyaránt található egy előrecsatolt háló (Feed Forward), amely a bemeneti adatot teljesen összekötött rétegen keresztül transzformálja, ezáltal növelve a szöveg reprezentációjának gazdagságát.

A végső kimeneti rétegek is eltérőek: míg a BERT főként osztályozási feladatokra van optimalizálva, addig a GPT szöveg-előrejelzést végez. A BERT így a nyelvi megértésben és szöveganalízisben nyújt kiemelkedő teljesítményt, míg a GPT elsősorban a szöveg generálásában jeleskedik, köszönhetően az autoregresszív működésének, amely lehetővé teszi számára, hogy fokozatosan építse fel a szöveget az előző szavak alapján. Az ábra tehát rámutat a két modell közötti különbségekre és hasonlóságokra, kiemelve, hogy a BERT és a GPT eltérő architektúrái hogyan szolgálják a nyelvfeldolgozás különböző céljait.

5.3 A BERT ÉS A TRANSFORMER ARCHITEKTÚRA

Mint már többször is említettem, a modell a transformer architektúra kódoló részét használja fel a működése során, amely többlépcsős feldolgozást alkalmaz a bemenetre. Az input egy tokenekből álló adatfolyam, mely a nyers szöveg alapján készül el. Ezekhez először beágyazási vektorokat rendelnek, majd a neurális hálózat feldolgozza ezen struktúrákat. Mielőtt azonban magát a folyamatot ábrázolnám, először kifejtem, hogy mi is maga a beágyazási vektor.

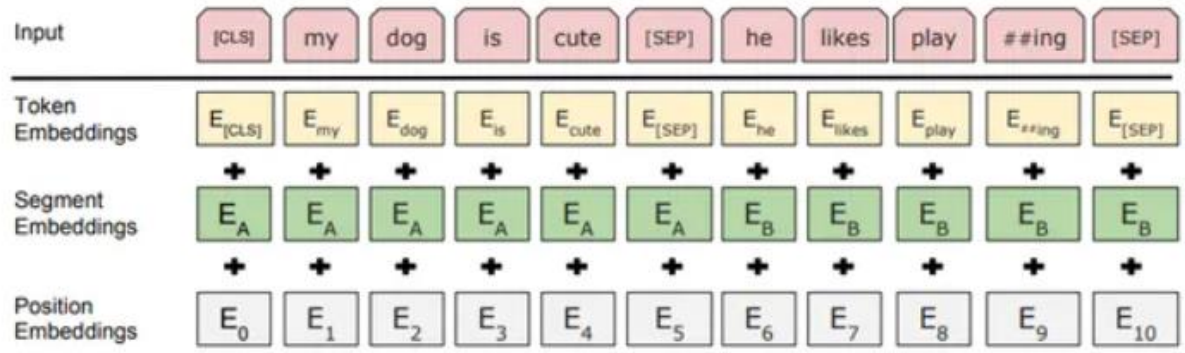
A beágyazási vektor a nyelvi tokenek numerikus megvalósulása egy többdimenziós térben, amely lehetővé teszi, hogy a neurális hálózatok értelmezni tudják a szövegeket. A cél az, hogy a szavakat vektorok formájában ábrázolni lehessen, ahol a hasonló jelentésű szavak vektorai egymáshoz közel helyezkednek el a térben, míg a különböző jelentésű szavak távolabb. Matematikai képletét a 7.ábrában láthatjuk:

$$E : V \rightarrow \mathbb{R}^d$$

7. ábra: Beágyazási képlet

Forrás: Vaswani, et al., 2017

Ahol a V a szókincs, E pedig egy olyan függvény, amely minden tokenhez egy d dimenziós valós számokból (R) álló vektort rendel.



8. ábra: Kódolási folyamat

Forrás: Kajal (2023)

A BERT bemenete ahogyan a 8.ábrán is láthatjuk, három komponensből áll:

- **Token beágyazás** (Token Embeddings),
- **Szegmens beágyazás** (Segment Embeddings),
- **Pozíciós beágyazás** (Positional Embeddings).

Az első lépésnél, tehát a token beágyazásnál, a bemeneti szöveget tokenekre bontja, majd minden egyes tokenhez egy egyedi vektort rendel, amelyeket majd a neurális hálózat feldolgoz. A BERT a WordPiece tokenizációt használja, melynek különlegessége, hogy a tokenek nem csak szavak, de szótöredékek is lehetnek. Az utóbb említett módszer ahelyett, hogy minden egyes szót külön tokenként kezelne, az egyes szavakat kisebb egységekre bontja. Ezek az egységek lehetnek gyakori szógyökök vagy elő- és utótagok, de előfordul, hogy adott egység egy teljes szót magában foglal. Példának okán a „megbocsáthatatlan” szót „megbocsát” és „##hatatlan” részekre bontana fel. Ez a megközelítés különösen hasznos ritka vagy ismeretlen szavak esetében, mivel nem szükséges, hogy a modell az egész szót ismerje. Ehelyett a gyakori szóalkotó részeket használja, amelyekből a modell értelmes reprezentációt tud alkotni. Így jobban tud általánosítani adott szöveg feldolgozásakor, és csökkenti az ismeretlen szavak miatti hibákat, amelyek egyébként megzavarnák a tanulást vagy a predikciót.

A második lépés a szegmens beágyazás. Mivel a BERT támogatja mondatok közötti kapcsolatok tanulását is, minden tokenhez hozzáad egy szegmens beágyazást, amely jelzi, hogy az adott token hanyadik mondathoz tartozik.

Mivel a transformer architektúra nem rendelkezik szekvenciális memória-struktúrával, a BERT-nek szüksége van egy megoldásra, hogy megtartsa a szavak sorrendi információját. Ehhez pozíciós beágyazási vektorokkat kell rendelni minden tokenhez, amelyek jelölik

valamennyi elhelyezkedését a szekvencián belül. A pozíciós kódolás matematikai képletét a 9. ábra mutatja:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

9. ábra: Pozíciós kódolás matematikai képlete

Forrás: (Vaswani, et al., 2017)

ahol *pos* a token pozíciója a szekvenciában, *i* a vektor dimenziójának az indexe, d_{model} pedig a beágyazási dimenzió, amely a BERT és a transformer modell belső dimenzióját adja meg. A pozicionális kódolás olyan vektorokat állít elő, amelyek hozzáadódnak a bemeneti tokenek beágyazásaihoz. Ezáltal a modell számára egyértelművé válik a tokenek sorrendje anélkül, hogy explicit módon megadnánk azt. A képletben az *i* index minden páros és páratlan pozícióhoz eltérő funkciót rendel. A páros indexeknél (2*i*) a pozícióhoz egy szinusz alapú kódolást alkalmaz (*sin*), míg a páratlan indexeknél (2*i*+1) egy koszinusz alapú kódolást (*cos*). Ez a képlet biztosítja, hogy a különböző pozíciókhoz tartozó vektorok numerikusan megkülönböztethetők legyenek, és a modell felismerje a szavak sorrendjét.

A BERT bemenete minden tokenre vonatkozóan a token beágyazás, a szegmens beágyazás és a pozíciós beágyazás összegeként áll elő. Ezeket a vektorokat együttesen dolgozza fel a BERT modell, hogy a szöveg jelentését, a mondatok közötti kapcsolatokat, valamint a szavak sorrendi információit figyelembe véve tudja végezni a nyelvi feladatokat.

A BERT modell a transformer architektúrának nem csak a kódoló komponensét használja, de az önfigyelési mechanizmusát és a teljesen kapcsolt hálózat komponenseket is. Az önfigyelési mechanizmus felel azért, hogy minden egyes tokent egy szövegen belül egyszerre figyelje, mindkét irányba egyaránt. Az önfigyelés lényege, hogy a modell egy súlyozott figyelmi mátrix (*A*) segítségével határozza meg, hogy egy adott szó mennyire fontos egy másik szó szempontjából. A figyelmi súlyokat a softmax függvénnyel skálázzák, amely biztosítja, hogy a súlyok összege 1 legyen. Ezt a képletet a 10. ábra mutatja be:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

10. ábra: Önfigyelési mechanizmus matematikai képlete

Forrás: (Vaswani, et al., 2017)

ahol *Q* a lekérdező mátrix, *K* a kulcs mátrix, *V* pedig az érték mátrix és d_k a kulcsok dimenziója, amely egy fontos skálázó faktor szerepét tölti be. A *softmax* függvény itt a

tokenek közötti fontosságot határozza meg. A *softmax* rész lényegében a kapott értékeket egy valószínűségi eloszlásra alakítja át, amely összege minden esetben 1 lesz. Ez a softmax eredményeként kapott érték a figyelmi súlyokat jelenti, amelyek azt mutatják, hogy az adott kérdés mennyire figyel egy adott kulcsra.

A többfejű figyelési mechanizmus lehetővé teszi, hogy a modell különböző szempontokat figyelembe véve vizsgálja meg a kapcsolatokat. Például szintaktikai és szemantikai összefüggések alapján is egyaránt. Az önfigyelési rétegek után minden egyes token keresztülmegy egy teljesen összekapcsolt, két rétegű Feed-Forward neurális hálózaton.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

11. ábra: Feed-Forward matematikai képlete

Forrás: (Vaswani, et al., 2017)

A képletben az x: a bemeneti vektor egy adott pozícióhoz, A W_1 és a W_2 a lineáris (teljesen összekötött) rétegekhez tartozó súlymátrixokat jelenti. A b_1 és a b_2 az első és második lineáris réteghez tartozó bias (eltolási) vektorokat képviselik. Az első lépésben x-et megszorozzuk az első W_1 súlymátrixszal és hozzáadjuk a b_1 bias vektort. Ez az átalakítás megváltoztatja a vektor dimenzióját. Ezt a dimenziónövelést azért végezzük, hogy nagyobb kapacitást biztosítsunk a hálózatnak, lehetővé téve a bemenetek komplexebb feldolgozását. Ezután a ReLU (Rectified Linear Unit) aktivációs függvényt alkalmazzuk, ami képletben a következő függvényként jelenik meg: max(0,z). Ez azt jelenti, hogy az összes negatív érték 0 lesz, míg a pozitív értékek változatlanok maradnak. A ReLU nemlinearitást vezet be, amely segít a modellnek bonyolultabb mintákat megtanulni. Az aktivációs függvény után az eredményt megszorozzuk a második súlymátrixszal W_2 majd hozzá adjuk a b_2 bias-t is. Ezzel visszatérünk a kezdeti dimenzióhoz, azaz a kimeneti dimenziója ugyan annyi lesz, mint a bemeneti. Ez a két lépéses, ReLU-val közrefogott lineáris átalakítás növeli a modell tanulási kapacitását. Mivel minden pozícióban külön-külön történik, az FFN lehetővé teszi a modell számára, hogy összetett mintákat tanuljon a bemenetekből, amelyek szükségesek a nyelvi összefüggések felismeréséhez. Az FFN hasonló szerepet tölt be, mint egy mélyebb neurális réteg a hagyományos mély tanulási modellekben.

A hálózat minden tokenre külön-külön alkalmaz egy lineáris transzformációt, majd egy nemlineáris aktivációs függvényt. Minden ilyen transzformáció után maradék kapcsolatok és réteg-normalizálás biztosítja a modell stabilitását és jobb konvergenciáját. A maradék kapcsolat azt jelenti, hogy az egyes rétegek kimenetéhez hozzáadják az adott réteg bemenetét, így:

$$y = F(x) + x$$

12. ábra: maradékkapcsolat matematikai háttere 1

Forrás: (Vaswani, et al., 2017)

ahol *x* az adott réteg bemenete, és *F(x)* az a transzformáció, amelyet az adott réteg végez. Ez segít megakadályozni az információ elvesztését a mélyebb rétegekben, és megkönnyíti a tanulást, különösen a mélyhálózatok esetén. A rétegnormalizálás biztosítja, hogy minden réteg kimenetének eloszlása egyenletesen skálázott és központosított legyen. Minden egyes réteg bemeneti adatainak átlagát és szórását normalizálják, így azok a következő rétegbe megfelelő skálázásban kerülnek be. Matematikailag:

$$\hat{x} = \frac{x - \mu}{\sigma}$$

13. ábra: Maradékkapcsolat matematikai háttere 2

Forrás: (Vaswani, et al., 2017)

ahol μ a bemenet átlagértéke, σ pedig a szórása. (Vaswani, et al., 2017)

Ezek a technikák együtt segítenek abban, hogy a BERT modell pontos és hatékony legyen a szövegek feldolgozásában, miközben biztosítják a tanulási folyamat stabilitását és gyorsaságát. A BERT modell továbbá pre-training és fine-tuning stratégiát használ az adatok értelmezéséhez, és a betanulási folyamathoz. Ezeket a következő fejezetben fogom részletezni.

6. A TANÍTÁSI FOLYAMAT BEMUTATÁSA

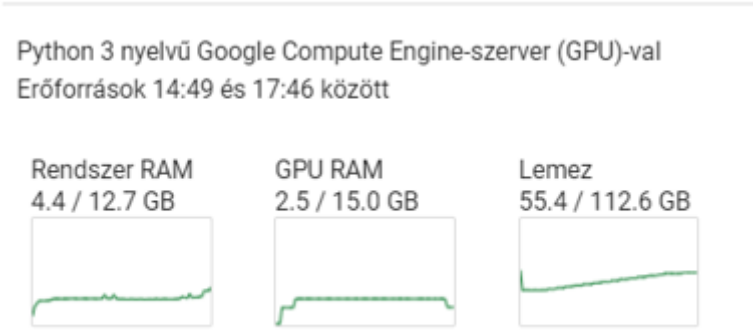
A modell tanítása két csoportra osztható, pre-trainingre, és fine-tuningra. A pre-training folyamat, ami a leginkább idő- és erőforrásigényes, ha egy ilyen modellt szeretne valaki létre hozni. Bár én ezzel a résszel nem foglalkoztam a munkám során, a teljes kép átlátása érdekében néhány mondatban ezt a folyamatot is bemutatom. Az a modell, amelyet én is használtam összesen 13 napig tartott, amíg betanult, azaz végzett a pre-training szakasszal.

Sok minden befolyásolhatja a betanítási időt, például a modell architektúrája, a korpusz mérete, vagy a használt hardver. Az alapvető BERT modellek általában néhány nap, vagy 1 hét alatt is be tudnak tanulni egy megfelelő GPU klaszteren. Az eredeti BERT Base modell betanítása 16 TPU-n körülbelül 4 napig tartott, míg a BERT Large modell ugyanennyi TPU-n 4-7 napon át tanult, attól függően, hogy milyen specifikus korpuszon futtatták. Utóbbi esetben az eltérés a korpusz méretétől és a hardverbeállításoktól függött leginkább.

A fine-tuning folyamat az, amivel én is dolgoztam, és amelynek az eredményeit a későbbiekben le is írtam. Ez a folyamat, átlagosan 1-3 órán át szokott tartani, egy modern,

nagy teljesítményű GPU-val, viszont, ha nagy az adatállomány, akkor ez akár 5-10 órán át is eltarthat. Egy olyan modell esetén, mint például a BERT Large pedig akár 12-24 óra is lehet, mivel több paraméterrel rendelkezik, mint egy alap modell. (Devlin, et al., 2019)

Az én esetemben a tanítási idő körülbelül 2 óra volt, viszont én egy BERT Base modellt használtam, azokkal a technikai beállításokkal, amelyek a 14. ábrán láthatóak:



14. ábra: Használt technikai hátté

Forrás: Google Colab, 2024

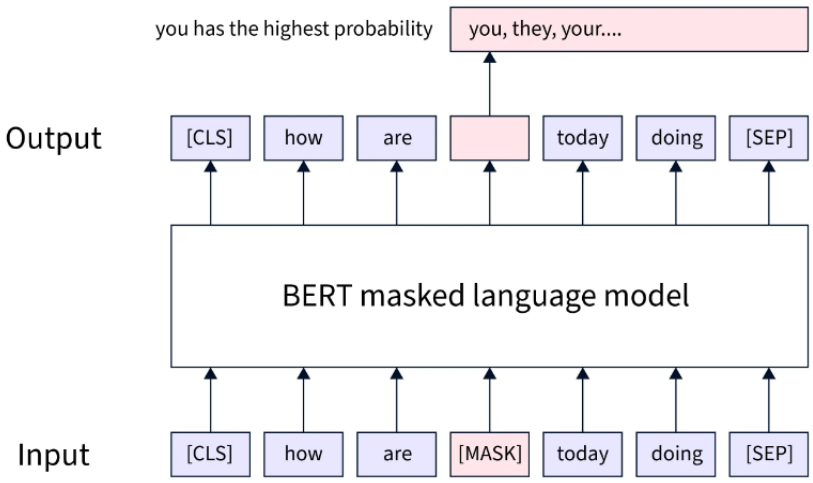
15 GB-os GPU-val dolgoztam, 12,7 GB RAM-al felszerelt rendszerrel. Ez elegendő volt ahhoz, hogy ezt a modellt finomhangolni tudjam, viszont a Colab ingyenes verziójával nem maradt utána sok időm a tesztelésre, hiszen kifogytam a Google által ingyenesen biztosított számítási egységekből. Kipróbáltam továbbá A100 típusú GPU-n hogy meddig tart a folyamat. Ebben az esetben sikerült a futási időt 1 órára csökkentenem.

6.1 PRE-TRAINING FOLYAMAT

BERT pre-training szakasza két fő feladatot foglal magában:

- **Maszkos nyelvmodellezés** (Masked Language Modeling - MLM),
- **Következő mondat előrejelzése** (Next Sentence Prediction - NSP).

A BERT leginnovatívabb része az MLM, amely lehetővé teszi a modell számára, hogy a szövegkörnyezetben két irányából ismerje meg a szavak jelentését. Ennek érdekében a bemeneti szöveg egy részét véletlenszerűen elrejtik (maszkolják), és a modell feladata ezeknek a szavaknak a predikciója.



15. ábra: MLM maszkolás

Forrás: Cathrine Jeeva, 2023

Az MLM működését vizuálisan a 15.ábra mutatja be, amely az alábbi folyamatot tartalmazza:

- **Maszkolás**: A bemeneti mondat tokenjeinek körülbelül 15%-át egy speciális [MASK] tokennel helyettesítik. Ez a token eltakarja az eredeti szót, amit a modellnek a szövegkörnyezet alapján kell visszaállítania. Fontos megjegyezni, hogy bár a szavakat maszkolják, a modell még mindig hozzáfér a szöveg teljes tartalmához, kivéve az eltakart részt.
- **Predikció**: A BERT modell célja, hogy a maszkolt szavakat kitalálja a fennmaradó szöveg alapján. Ehhez a modell a szókincsére (pre-trained vocabulary) és a kétirányú szövegkörnyezet értelmezésből származó kontextuális információkra támaszkodik.

Az MLM folyamat matematikai modellje nagyon összetett. Jelöljük az adott szöveget az alábbi adathalmazon keresztül: $X=[x_1, x_2, \ldots, x_m]$, ahol x_m az a token, amit maszkoltunk. A modell megbecsüli annak a valószínűségét, hogy az adott maszkolt token (x_m) milyen értéket vesz fel a többi token alapján. A becslés alapját a 16. ábrán látható képlet mutatja:

$$P(x_m|X \setminus m) = \text{softmax}(W_h h_m)$$

16. ábra: MLM matematikai háttere 1

Forrás: Cathrine Jeeva, 2023

ahol $X \setminus m$: a bemenet a maszkolt token nélkül, h_m: a maszkolt token rejtett állapota, amit a transformer encoder előállít, W_h: súlymátrix a kimeneti réteghez, softmax: aktivációs függvény, amely az egyes tokenek valószínűségét adja meg.

A BERT modell a fenti folyamat során egy veszteségi függvényt minimalizál, amivel maximalizálja a helyes tokenek előrejelzésének valószínűségét. A veszteségi függvényt a 17. ábrán látható képlet írja le:

$$L_{\text{MLM}} = -\sum_{m} \log P(x_m | X_{\setminus m})$$

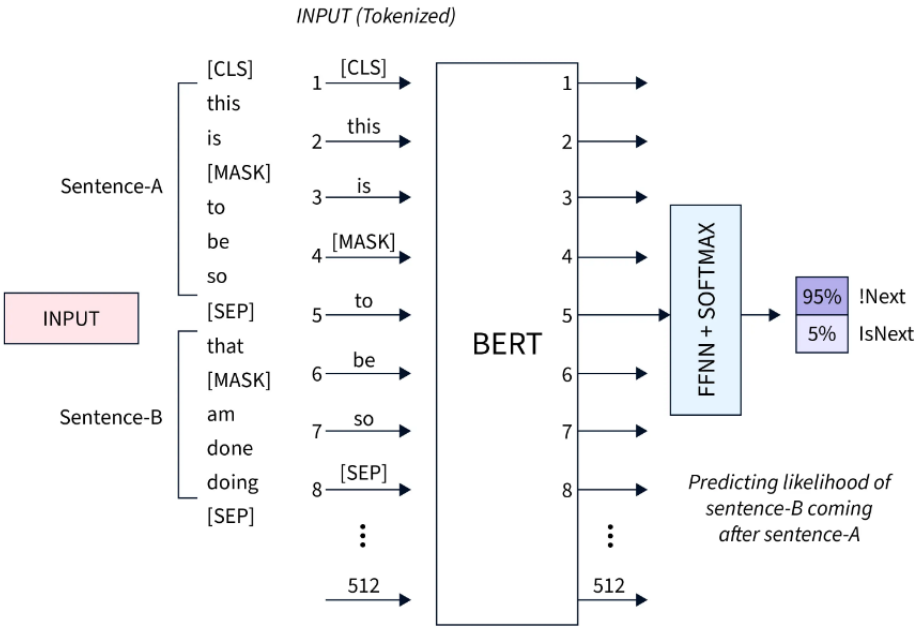
17. ábra MLM matematikai háttere 2

Forrás: Cathrine Jeeva, 2023

Ez a veszteségi függvény azt biztosítja, hogy a modell a tanulási folyamat során a helyes szavak előrejelzésére optimalizáljon. Például az input mondat "*A macska alszik az ágyon*", és a "*macska*" szót maszkoljuk, akkor az a cél, hogy a modell próbálja a "[MASK]" helyére a "*macska*" szót helyesen bejósolni a környezeti információk alapján *"A [MASK] alszik az ágyon."*. (Liu et al., 2019)

A BERT második fontos pre-training feladata az NSP, amely azt vizsgálja, hogy két mondat logikailag összefügg-e egymással, vagy véletlenszerűen kerültek egymás mellé. A tanulási folyamat során a modell bemeneteként két mondatot kap. Az esetek 50%-ában a második mondat valóban az első mondat után következik az eredeti szövegben, míg a fennmaradó 50%-ban a második mondat véletlenszerűen lett kiválasztva egy másik szövegből, azaz nem kapcsolódik az első mondathoz. A modellnek meg kell tanulnia, hogy mely mondatpárok kapcsolódnak logikailag egymáshoz, és melyek nem. Az NSP (Next Sentence Prediction) két fő részre osztódik:

- Mondatok közötti kapcsolat felismerése: A BERT modell az előző mondat kontextusát használja annak megállapítására, hogy a második mondat természetesen követi-e az elsőt. Ez különösen fontos, mivel a mondatok közötti logikai és szerkezeti kapcsolatok kulcsszerepet játszanak a szöveg globális megértésében.
- Fontosság megállapítása: Ez a második komponens azt vizsgálja, hogy a második mondat mennyire fontos az első mondat szempontjából. A BERT-nek meg kell tanulnia felismerni, hogy a második mondat hogyan kapcsolódik az előző mondatban található információkhoz, hogy mélyebb összefüggéseket hozzon létre a szövegben.



18. ábra: NSP működése,

Forrás: Cathrine Jeeva, 2023

Az NSP folyamatát a 18.ábra mutatja be, melynek lépései:

- **Speciális tokenek használata**: Az első mondat elején egy [CLS] (*classification - osztályozó*) token található, és minden további mondat végén egy [SEP] (*separator - elválasztó*) token kerül elhelyezésre.
- **Mondatbeágyazás**: Minden tokenhez hozzáadnak egy mondatbeágyazási vektort, amely jelzi, hogy az adott token az *A* vagy *B* mondathoz tartozik. A mondatbeágyazás segítségével a modell meg tudja különböztetni az első és a második mondathoz tartozó tokeneket.
- **Pozíciós beágyazás**: Minden token kap egy pozíciós beágyazást, amely azt jelzi, hogy hol helyezkedik el a token az adott szekvencián belül. Ez fontos, mert a transformer architektúra önmagában nem képes figyelembe venni a tokenek sorrendjét, így a pozíciós információt explicit módon kell biztosítani.
- **Kimeneti réteg és döntés**: A feldolgozás után a BERT modell egy teljesen összekapcsolt (feed-forward) neurális hálózatot és egy softmax függvényt használ annak érdekében, hogy megbecsülje a valószínűségét annak, hogy a második mondat logikailag az első után következik-e vagy sem. A modell végül egy kétértékű döntést hoz*: „Next”* (azaz a második mondat valóban követi az elsőt) vagy *„Not Next”* (azaz véletlenszerű mondat).

A fentiek alapján jól látható, hogy a BERT modell előtanítása összetett és időigényes folyamat, amely számos bonyolult lépést foglal magában. Azonban ennek köszönhetően egy olyan modellt kapunk, amely kiválóan képes értelmezni a mondatok közötti kapcsolódásokat és összefüggéseket.

6.2 FINE-TUNING FOLYAMAT

A fine-tuning során a modell minden paraméterét újra optimalizálják, nem csak a hozzáadott kimeneti réteget. Ez lehetővé teszi, hogy a modell megőrizze az előzetes tanulás (pre-training) során szerzett általános nyelvi ismereteket, miközben alkalmazkodik egy adott feladat specifikus követelményeihez és mintázataihoz. A fine-tuning során gyakran kisebb tanulási rátát alkalmaznak, mint a pre-training fázisban. A tanulási ráta meghatározza, hogy a modell paraméterei milyen mértékben változnak egy-egy lépés során a gradiens alapú optimalizációban. A fine-tuning folyamat során tipikusan az értéke 1×10^{-5} és 5×10^{-5} közé esik. A kisebb tanulási ráta lehetővé teszi, hogy a modell finomabban igazítsa a paramétereit, elkerülve, hogy túlzottan megváltozzanak az előzőleg szerzett nyelvi képességei. Ez a fokozatos tanulási folyamat csökkenti a túlilleszkedés esélyét is. (Mosbach, et al., (2021))

Mivel a fine-tuning folyamata kisebb adatállományon történik, mint a pre-training, így a felhasználók gyakran 10-15 epoch alatt finomhangolják a modellt. Az epoch azt jelenti, hogy a modell egyszer végigmegy a teljes tanító adathalmazon. Minél több epochon keresztül tanítják a modellt, annál több lehetősége van a súlyok finomítására, de túl sok epoch túlilleszkedéshez vezethet, ami azt jelenti, hogy a modell túlságosan alkalmazkodik a tanító adatokhoz, és nem tud jól általánosítani az új adatokra.

A fine-tuning során szükség van az eredmények folyamatos ellenőrzésére és az esetleges túlilleszkedés elkerülésére. A validációs adatok alapján történő kiértékelés segít a modell teljesítményének nyomon követésében, és szükség esetén a hiperparaméterek finomhangolásában, például a tanulási ráta módosításában vagy a csoportméret beállításában. Az optimalizáció során gyakran alkalmaznak olyan technikákat, mint a korai leállás, hogy elkerüljék a túlilleszkedést, valamint dropout rétegeket a túltanítás minimalizálása érdekében. A korai leállás lényege, hogy ha a modell validációs teljesítménye több epoch után sem javul, a tanulási folyamatot leállítják. Ez segít abban, hogy a modell ne igazodjon túlzottan a tanító adatokhoz. A dropout pedig egy másik technika, amely véletlenszerűen kikapcsol bizonyos neurális kapcsolatokat a tanulás során, ezáltal szintén csökkentve a túlilleszkedés valószínűségét. A dropout lehetővé teszi, hogy a modell robusztusabbá váljon, mivel nem engedi, hogy a modell túlságosan függjön egyes kapcsolatoktól vagy neuronoktól. (Srivastava, et al., 2014)

7 ESETTANULMÁNY

7.1 A TANÍTÁS SORÁN HASZNÁLT KÖRNYEZET

Már többször is említettem, hogy az egyik legnagyobb kihívást számomra az erőforrás igények jellentették, mivel a saját személyes számítógépemen keresztül hajtottam végre magát a feladatot, amelynek a kapacitásai minimális erőforrást tudtak csak nyújtani. Mivel a modell tesztelései, illetve finomhangolása folyamatán jelentős tényező volt az idő, emiatt egy alternatív megoldást kerestem. Több lehetőséget is számításba vettem, de a legoptimálisabb rendszernek a Google Colaboratory-t találtam.

A Google Colaboratory (röviden: Google Colab) egy ingyenes, felhő alapú szolgáltatás, amely a gépi tanulás, adatfeldolgozás és egyéb programozási feladatok végrehajtására használható. Ezt a Google hozta létre, és a Jupyter Notebook alapú környezetet biztosítja, hogy a felhasználók böngészőjükben futtathatnak Python kódot, a Google szerverein anélkül, hogy helyileg kellene telepíteniük bármilyen szoftvert. Nagy előnye továbbá, hogy van GPU és TPU támogatása is, amelyek a gépi tanulási modellek futtatásának és tréningezésének a gyorsabbá tételét teszik lehetővé. Az ingyenes verzióban is elérhető ez a két lehetőség, viszont hátránya, hogy csak bizonyos ideig használható egy adott rendszer, ami miatt könnyen megszakadhat a futási folyamat. Mivel nekem fontos volt, hogy hosszabb időn keresztül tudjam tanítani, illetve tesztelni a modellt, emiatt én a Colab Pro verzióját használtam.

A rendszer legelőnyösebb része az volt, hogy itt sokkal gyorsabban tudtam finom hangolni a modellt, mit a személyes számítógépemen, majd a folyamat után a kész modellt el tudtam menteni és a későbbiekben fel tudtam használni további kiértékelésekre és tesztelésekre.

7.2 A TANÍTÁS SORÁN HASZNÁLT ADATHALMAZOK ÉS AZOK JELLEMZŐI.

A tanítás során felhasznált adathalmazt egy Excel-táblázatban kaptam meg, amely sajnos tisztítatlan volt, ezért szükséges volt először alapos adatellenőrzést és tisztítást végeznem, mielőtt a modell betanítását megkezdhettem volna. Az Excel fájl több mint 45 ezer terméket tartalmazott, amelyekről a következő információkat rögzítették: cikkszám, terméknév, gyártói cikkszám, gyártó neve, valamint a termék főcsoportja és két alcsoportja. Fontos megjegyezni, hogy a táblázat oszlopai hierarchikus struktúrába voltak szervezve. A "főcsoport" képviselte a legmagasabb kategóriát, míg az "alcsoport" és "alcsoport2" a termékek alacsonyabb szintű kategorizálását jelentették, melyen belül az „alcsoport” képviselte az általánosabb, magasabb szintű kategória osztályt. Mivel a termék adatokat manuálisan rögzítették az évek során,

kiemelt fontosságú volt az adatok pontosságának és teljességének ellenőrzése, hiszen emberi hiba mindig előfordulhat az ilyen jellegű folyamatokban.

Különös figyelmet fordítottam a termékek cikkszámára, mivel ez számított egyedi azonosítónak, így elengedhetetlen volt a pontos adatkezeléshez. A feldolgozást a pandas könyvtár segítségével kezdtem el, amellyel a Jupyter notebook környezetben olvastam be az adatokat. Első lépésként ellenőriztem, hogy a termékek cikkszámában van-e duplikáció. Mivel a cikkszám minden termék esetében egyedi kell, hogy legyen, ez kritikus szempont volt, de szerencsére nem találtam duplikált cikkszámokat, így folytathattam az adatok további tisztításával.

Az adatok nem minden esetben voltak teljesek. Különösen az „alcsoport” oszlopban voltak hiányos mezők, amelyeknél a „alcsoport2” oszlop már tartalmazta a szükséges információkat. Az ilyen esetekben az „alcsoport” és „alcsoport2” megegyezett volna, de csak az utóbbi oszlopot töltötték ki, ezért a pandas segítségével azokat a sorokat kerestem meg, ahol az „alcsoport” üres volt, és a „alcsoport2”-ben található érétket átmásoltam az „alcsoport” oszlopba. Ezt egy egyszerű *loc* függvény használatával gyorsan és hatékonyan meg tudtam oldani, automatizálva az adatpótlási folyamatot.

A tisztítási folyamat tehát magába foglalta a duplikációk ellenőrzését, valamint az adatok kiegészítését az „alcsoport” oszlopban.

7.3 A TANÍTÁSI FOLYAMAT ÉS A HASZNÁLT BERT-MODELL

Mielőtt elkezdtem volna bármit is a kész adatokkal, előbb kikellett választanom a megfelelő modellt hozzá. Rengeteg féle BERT-modell létezik, mindegyik más és másra specializálva, fejlesztve. Mivel az adatbázisom nem tartalmazott nagyon bonyolult szövegeket, és a célom a kategorizálás volt, így egy kisebb is megfelelt számomra. Ez biztosította továbbá azt is, hogy a modell minél gyorsabban fel tudjon állni, hiszen egy nagyobb modell betanítása sokkal időigényesebb lett volna. További szempont volt az is, hogy maguk az adatok magyar nyelven voltak írva, emiatt problémás lett volna egy olyan modellt használni, amely nem tudja jól kezelni a különféle ékezetekkel rendelkező szavakat és betűket. Két féle BERT modell jött szóba a célom elérése érdekében, az egyik az mBERT (Multilingual BERT), a másik pedig a HuBert (Hungarian language BERT).

Az előbbit azért fejlesztették ki, mivel az eredeti modellt angol nyelvre specializálták, és szerettek volna egy olyan modellt is, amely hasonlóan tud teljesíteni más nyelveken is, nem csak az angolon. Az utóbbi modellt pedig egy magyar informatikus fejlesztette ki, amelyet a saját anyanyelvére taníttatott be. Bár mindkét modellt felállítottam a kutatásom során, a magyar nyelvű modell sokkal jobban teljesített a kategorizálás során. Arra következtettem,

hogy ezért lehetett, mert az én adathalmazom erőteljesen használta a magyar nyelvet, és ezt a HuBERT jobban kezelte, mivel az előzetesen tanult folyamata során is kizárólag magyar nyelvű szövegeket használtak. Ennek következtében sokkal gyorsabban és sikeresebben fel tudta ismerni a szavak közötti kapcsolatot, így a későbbiekben is ezt a modellt elemeztem ki részletesebben.

7.3.1 HUBERT (HUNGARIAN LANGUAGE BERT MODELL)

Nemeskey Dávid Márk 2020-ban kifejlesztette a HuBERT modellt, amely egy magyar nyelvre specializált BERT Base implementáció. Itt fontos megemlítenem, hogy két féle HuBERT létezik, viszont a kettő nem összekeverendő. Amit én használtam HuBERT, a magyar nyelvre lett specializálva, míg a másik HUBERT modell, amelynek teljes neve Hidden Units BERT viszont hang alapú elemzésekhez lett kifejlesztve.

A magyar nyelv különböző jellegzetességei miatt nem volt optimális az angol nyelvre tanított modell használata a magyar nyelvű terméknevek értelmezésére. A nyelvtani szabályok bonyolultsága, főleg a toldalékolt szavak felismerése, a szabad szórendiség, a nagyon komplex ragozási rendszerünk miatt az angol nyelvű modellek általánosságban nem tudtak jó eredményeket produkálni a magyar szövegekben. Ennek okán lett létrehozva ez a modell, amely ezen sajátosságokra lett kiképezve, így jelentősen jobb eredményeket tudott elérni

A magyar HuBERT alapja egy BERT-Base modell, amelyet már korábban részletesen ismertettem. Felépítésében és architektúrájában is ugyan úgy működik, mint minden más modell, egyedül az elő tanítási adatai különböztek a többi modelltől. Ugyan is a HuBERT modellt nagyméretű, magyar nyelvű szövegeken képezték, hogy pontosan illeszkedjen a magyar nyelv jellemzőihez. Ezek a szövegek pedig a Magyar Wikipédiából, Közigazgatási dokumentumokból, illetve közösségi médiákból és webes szövegekből származtak.

7.3.2 A MODELL TANÍTÁSA

A legelső lépésem az volt, hogy betöltöttem azokat a Python könyvtárakat, amelyek szükségesek a modell felépítéséhez és betanításához. Ezek közül az egyik legfontosabb a *transformers* könyvtár volt, amelyet a HuggingFace szervezet biztosít a BERT modellek kezeléséhez. Ezzel a könyvtárral könnyen dolgozhattam előre betanított modellekkel, csak úgy, mint a HuBERT-el is. A pandas könyvtárat szintén importáltam, mert szükség volt rá az adatok kezeléséhez, különösen az Excel fájlban tárolt termékinformációk beolvasásához és feldolgozásához.

Miután a tisztított és nyers adatokat beimportáltam a környezetbe, kiválasztottam azokat az oszlopokat, amelyekkel dolgozni szerettem volna. Kizárólag a termék neveket (cikk_nev), a fő termékkategóriát (focsoport) és az alcsoportokat (alcsoport, alcsoport2) használtam. A

termék neve volt a bemenet, amit a modellnek osztályoznia kellett, míg az alcsoportok voltak azok a címkék, amelyek alapján a modell megtanulta a termékeket kategorizálni.

A következő lépésben megvizsgáltam az adatokat, és csoportosítottam őket az alcsoportjuk szerint. Itt rájöttem, hogy vannak olyan kategóriák, amelyek csak egyszer fordulnak elő az adathalmazban. Ezeket az alcsoportokat külön választottam, még a tanító és teszt halmaz szétválasztása előtt, mivel a tanítási folyamat egyik alap feltétele volt, hogy a tanító adathalmaznak mindenképpen tartalmaznia kellett az összes alkategóriát. Így miután ezeket a sorokat eltávolítottam, az adatokat szétválasztottam tanító és teszt halmazokra. Ezután azokat a sorokat, amiket eltűntettem vissza illesztettem a tanító adathalmazba, hiszen ezen kategóriák létezéséről is tudomást kell szereznie a modellnek, hogy újonnan hozzáadott termékeknél adott esetben ide is csoportosíthassa őket.

A tanító és teszt adatok arányainak meghatározása az egyik legfontosabb tényező egy modell tanítása előtt. Ha túl sok adat kerül a teszt adatbázisba, akkor előfordulhat, hogy a modell nem megfelelően fog betanulni, és nem a célnak megfelelően fog működni. Ha viszont túl sok adat kerül a tanító adathalmazba, akkor a finomhangolási folyamat fog nagyon hosszadalmassá válni. Ennek okán a szakirodalmat követve a teljes adatmennyiség 80%-át választottam ki tanítási adathalmaznak, mivel véleményem szerint ez egy jó arány ahhoz, hogy a modell megfelelő alapossággal be tudjon tanulni, viszont a folyamat sem lesz nagyon lassú.

A szétválasztást követően még egy fontos lépést hajtottam végre, a tanító halmazok sorait összekevertem. Ez a lépés bár elhanyagolhatónak tűnik, mégis jelentős a modell megfelelő finomhangolása érdekében. A legelső BERT modell publikációjában is kiemelték, hogy mivel a modell egy előzetesen betanított nyelvi modell, ha az adatokat nem keverjük össze, és azok csoportosítva kerülnek bemutatásra (pl. egy bizonyos osztályhoz tartozó adatok egymás után), akkor a modell könnyen túlilleszkedhet az adott osztály mintáira. Ez rosszabb általánosítási képességet eredményezhet a különböző osztályokkal szemben. Az összekevert adatok biztosítják, hogy a modell a tanulási folyamat során változatos mintákkal találkozzon, így jobban képes lesz általánosítani az adatok különböző jellemzőire, és nem csak egy adott osztály sajátosságaira. (Devlin et al., 2019). Így összességében elmondható, hogy ez a lépés ugyan olyan jelentőséggel rendelkezik, mint a tanítási rész megfelelő beállítása.

A szétválasztási folyamat után az adatok címkézése következett, melynek során elkülönítettem a címkéket és a szöveges adatokat. A címkék, amelyek az alcsoportok nevét tartalmazták, szöveges formában voltak, és ezeket a modell nem tudta közvetlenül kezelni. Ezért a címkéket számmá kellett alakítanom, amit a LabelEncoder segítségével valósítottam

meg. A LabelEncoder először megtanulta a különböző alcsoportokat (ez az úgynevezett "fit" művelet), majd minden alcsoportot átalakított egy numerikus értékké (ez a "transform" művelet). Így a modell számára már használható számszerű címkék álltak rendelkezésre.

A szöveges adatokat, azaz a termékek neveit, elő kellett készítenem a BERT modell számára, amihez a BERT tokenizálót használtam. A tokenizálás az a folyamat, amikor a szövegeket a modell által értelmezhető formátumra alakítom. A BertTokenizer segítségével ezt megtehettem úgy, hogy a szövegeket tokenekre bontotam, és biztosítottam, hogy mindegyik token azonos méretű legyen, hiszen ez szükséges volt ahhoz, hogy a modell megfelelően tudjon velük dolgozni. Itt használtam a SZTAKI-HLT/hubert-base-cc tokenizálót, amely kifejezetten magyar nyelvű szövegekhez készült, így ideális választás volt az én adathalmazomhoz.

A tokenizált adatokat és a címkéket egy egyedi ProductDataset osztályba csomagoltam, amely a torch.utils.data.Dataset osztályból örökölt. Ez az osztály tette lehetővé, hogy a tokenizált adatok és címkék megfelelő formátumban kerüljenek a PyTorch könyvtárba, amely a BERT modell futtatásához volt szükséges. Ebben az osztályban definiáltam a getitem metódust, amely biztosította, hogy az adatok és a címkék tensorok formájában legyenek visszaadva, ami azt jelenti, hogy a GPU vagy CPU tudja feldolgozni őket.

Miután létrehoztam a tanító és teszt halmazokat tartalmazó ProductDataset példányokat, ellenőriztem, hogy rendelkezésre áll-e GPU. Ez azért fontos, mert az ilyen mélytanulási modellek, mint a BERT, rendkívül számításigényesek, és egy GPU használata jelentősen megnöveli a tanítás sebességét. Mivel a Colab rendszerében dolgoztam, itt volt elérhető GPU, amit ki is használtam a folyamat gyorsítása érdekében.

A BERT modell inicializálása során a BertForSequenceClassification osztályt használtam. Ez az osztály kifejezetten szövegosztályozásra készült, és az általam betöltött modell bár már előre be volt tanítva, az én adataimra kellett finom hangolni. Meghatároztam, hogy hány különböző címkét kell a modellnek osztályoznia, tehát hány különböző kategória létezett az adathalmazomban. A modellt ezt követően a GPU-ra helyeztem át, hogy a tanítás hatékonyabb legyen.

Ezután meghatároztam a tanítási paramétereket a TrainingArguments segítségével. Ezek a paraméterek számos dolgot szabályoznak: például, hogy a tanítási folyamat során hol tárolja a modell a kimeneteket, hányszor értékelje ki a modellt, és hány epochon keresztül fusson a tanítás. A megfelelő epochok számának beállítása szintén egy fontos tényező volt, viszont nagyon időköltségesnek tűnt, hogy egyesével lefuttassam a modellt, különböző epoch számokra. Így úgy döntöttem, hogy a tanítási folyamatba beépítek egy korai leállás nevű

funkciót, ami a tanítási folyamat közben mérte a modell teljesítményét, és ha bizonyos számú egymást követő tanítási cikluson át nem javul az eredménye (az én esetemben ez 3 volt), akkor a tanítást megállítja. Ez azért tűnt optimális megoldásnak, mert ha a modell túl kevés epochon keresztül tanul, akkor alul tanult marad, és nem lett volna képes kihozni magából a maximális precizitást. Ha viszont túl sok tanítási körön ment volna keresztül, akkor fent állt volna a veszély, hogy a modell túltanult lesz, ami miatt szintén nem teljesített volna megfelelően, csak a tanító adathalmazon.

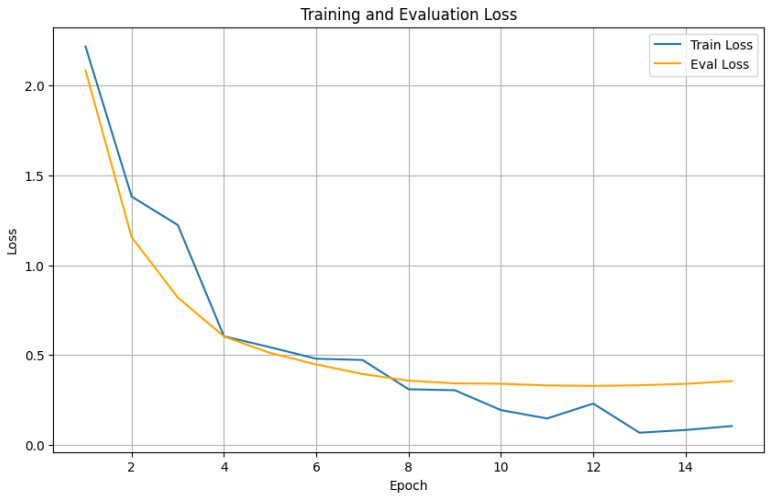
A tanításhoz a Trainer osztályt használtam. Először példányosítottam, majd a train() függvényének meghívásával elindítottam a modell tanítását. E folyamat során minden epoch után megjelent az aktuális veszteség, amely az epochok előrehaladtával fokozatosan csökkent. A korai megállás funkció általában a 15. epoch környékén állította le a folyamatot, ekkora sikerült a finomhangolás olyan szinten, hogy a modell a cikkek nevei alapján igen magas pontossággal volt képes kategorizálni a termékeket.

A finomhangolási folyamat legvégén elmentettem a modellt, és az általa használt transformereket azért, hogy ha későbbiekben tesztelni szerettem volna, akkor elő tudjam venni, és használni. Ennek okán nem kellett a modellt állandóan újra tanítanom, hanem a lementett modellt beimportálni, majd felhasználni. Ezzel a módszerrel jelentősen megkönnyítettem a tesztelési folyamatot, és egyúttal lehetővé tettem azt is, hogy ha későbbiekben kategorizálásra szeretné valaki használni, akkor egyszerűen csak behívja a modellt, és már tudja is használni.

7.4 EREDMÉNYEK, ÉS AZOK BEMUTATÁSAI

Mindezek után, ebben a fejezetben részletesen ismertetem a modell által elért eredményeket. Első sorban a finomhangolási folyamat eredményeit mutatom be, hogy ezen folyamat alatt, milyen sikerességgel tudott kategorizálni a modell.

Ehhez segítségül a 19. ábrát hoztam létre, amely vizuálisan mutatja be a tanításnak a folyamatát. Az ábrán láthatjuk a tanulási veszteség (Trainin Loss) és az érvényesítési veszteség (validation loss) mutatókat, melyek segítségével láthatjuk a modell teljesítményét és általánosító képességét az epoch-ok előrehaladtával. Látható, hogy a modell 14 epoch-on keresztül tanult, hogy el tudja érni a maximális teljesítményét.



19. ábra: HuBERT tanítási folyamat

Forrás: Sajátszerkesztés 2024

A grafikonon két fő görbe látható. A kék vonal mutatja a tanulási veszteséget, amely azt mérte, hogy mennyire illeszkedett a modell a tanító adatokra. Ennek a mutatónak az értékei a tanító adatokon elkövetett hibák méretét mutatják meg. Ez ideális esetben az epoch-ok előrehaladtával csökken, amely azt jelzi, hogy a modell egyre jobban illeszkedik a tanító adatokhoz. Látszódik is az ábrán, hogy az első 4. epoch során nagyon gyorsan csökkent ez az érték, amely azt bizonyítja, hogy milyen gyorsan megtanulta a modell az adatok mintázatait.

A másik görbét narancssárga színnel jelöltem, amely az érvényesítési veszteséget mutatta be. Ez a mutató mérte, hogy mennyire képes a modell az ismeretlen, validációs adatokon, jól teljesíteni. A grafikon alapján látható, hogy ez a mutató is gyorsan csökkent az első néhány epoch során, majd körülbelül a 7-8. epoch után ez a csökkenés lelassult. További részletes adatokat a 2. táblázat mutatja meg.

2. táblázat: Tanítási folyamat

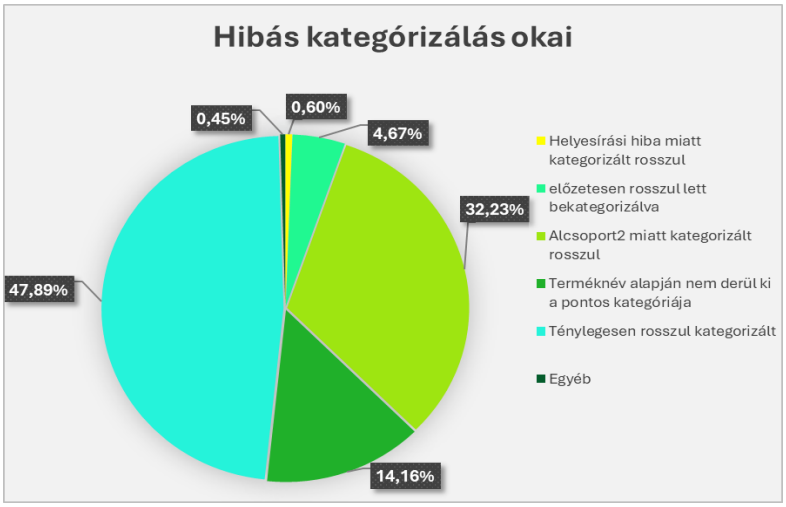
Epoch	Training Loss	Validation Loss
1	2.214900	2.082632
2	1.380700	1.154777
3	1.223000	0.819945
4	0.604700	0.603637
5	0.543900	0.512173
6	0.479900	0.448246
7	0.473300	0.395338
8	0.310500	0.358192
9	0.305100	0.343217
10	0.194600	0.341037
11	0.148200	0.331709
12	0.230900	0.329254
13	0.069100	0.332639
14	0.084200	0.340519

Forrás: Sajátszerkesztés, 2024

Érdekes megfigyelni, hogy az első epochnál a tanulási veszteség értéke a legmagasabb, ami 2,2149, ez azonban néhány epoch után jelentősen csökkent, a 11. epochnál például 0.1482-re, majd a 14. epochnál 0,0842-es végeredménnyel zárult. Ez arra utal, hogy a modell már nagyon jól illeszkedett a tanító adatokra. Érvényesítési veszteség is hasonlóan csökkent, mint az előbb bemutatott tanítási veszteség, viszont az itt látható értékek kicsit magasabbak lettek összességében. A 7-8. epochnál az érvényesítési veszteség csökkenése lelassult, és itt pontosan is láthatjuk, hogy e között a két epochnál volt meg a két mutató közötti metszet.

Mind a két esetben észre lehet venni, hogy az utolsó 3 epoch során kis mértékben nőtt a veszteségek értéke, amely a korábban említett korai leállás aktiválását váltotta ki. Ennek hasznossága itt mutatkozik meg, hiszen nem kellett megvárni, még akár több tíz epoch lefutását, hogy az összes eredmény birtokában önállóan döntsem el hanyadik epochnál jött létre a legjobban használható modell, hanem a korai megállás a tartós eredmény stagnálás miatt leállította a finomhangolás folyamatát egy megfelelően mértékben betanított modellnél.

Végül a teszt adatokon külön még egyszer lefuttattam a modellt, amely predikció során a modell összesen 93,66%-os pontossággal tudta jól bekategorizálni az adatokat. Mivel kíváncsi voltam, hogy mely termékek esetén nem sikerült a modellnek jó predikciót felállítani, a teszt adatokat külön elmentettem egy Excel táblázatba, hogy a későbbiekben ezeket tudjam elemezni. Itt sajnos nem találtam jobb módszert, mint hogy egyesével végig mentem a termékek nevein, majd felvázoltam a hibás kategorizálásnak a lehetőségeit, melyet egy diagrammon foglaltam össze. Ezt a diagrammot a 20. ábrán láthatjuk:



20. ábra: Hibás kategorizálás okai diagramm

Forrás: Sajátszerkesztés, 2024

A jelmagyarázatban jól látható, hogy az adott kategorizálási hiba okok milyen aránnyal fordultak elő. A helyesírási hibába azok az adatok kerültek, amelyeket a modell jó kategóriába sorolt be, viszont az eredeti kategóriájában helyesírási hiba volt található, és emiatt kerültek a hibásan kategorizált termékek közé.

3. táblázat: Példa a helyesírási hibákra

Cikk neve	**Hibás kategória**	**Jó kategória**
Flex DD 2G 18.0 akkus fúró csavarozó szett	Fúró-csavarozó	Fúrócsavarozó

Forrás: Sajátszerkesztés, 2024

Például ahogyan a 3. tábla is mutatja, ezt a terméket, a *Flex DD 2G 18.0 akkus fúró csavarozó szett*-et a modell a *Fúró-csavarozó* alcsoportba kategorizálta be, a helyes kategóriája pedig a *Fúrócsavarozó* volt. a két csoport között csupán annyi a különbség, hogy az egyiknél kötőjellel írták a kategória nevet, a másiknál meg anélkül.

Az „*előzetesen rosszul lett bekategorizálva”* csoportba azok kerültek, amelyek a tanítási folyamat előtt már rossz kategóriákba voltak sorolva. Ez az érték elképzelhető, hogy alacsonyabb, mivel én sem ismertem teljesen a termékeket, így csak valószínűsítettem melyek voltak azok, amelyeket a manuális folyamat során rosszul kategorizáltak. Például ide soroltam a, mint *Makita 4329JX4 szúrófurész* nevű terméket, amelyet eredetileg az *Elektromos dekopírfűrész*-hez soroltak, viszont a modell a *Szúrófűrész*-hez sorolta. A nevéből arra következtettem, hogy nem a modell tévedett.

4. táblázat: Példa az előzetesen rosszul lett kategorizálva csoportra

Cikk neve	**Hibás kategória**	**Jó kategória**
Makita 4329JX4 szúrófurész	Szúrófűrész	Elektromos dekopírfűrész

Forrás: Sajátszerkesztés, 2024

Az elemzés során az a hipotézis fogalmazódott meg bennem, hogy azért kategorizált rosszul a modell, mert az adattisztítási folyamat során néhány esetben az alcsoport2 kategóriái kerültek az alcsoport elemei közé[4]. Mivel előfordulhatott olyan termék is, amelyet bár eredetileg nem jó alcsoportba kategorizált, viszont az alcsoport2 kategóriáját jól be tudta azonosítani, így ezeket a termékneveket külön megjelöltem és az „*alcsoport2 miatt kategorizált rosszul*” csoportba soroltam őket. Ilyenre példát az 5. táblázat mutat be:

5. táblázat: Példa az alcsoport2 kategorizálásra

Cikk neve	**Hibás kategória**	**Jó kategória**	**Alcsoport2**
ORIT 3300 SS kézi kőroppantó	Kézi kőroppantó	Kőroppantó	Kézi kőroppantó
TR-6,5 L robbanómotoros háromfázisú aggregátor	Dízelmotoros háromfázisú aggregátor	Háromfázisú áramfejlesztő	Dízelmotoros háromfázisú aggregátor

Forrás: Sajátszerkesztés, 2024

Véleményem szerint ezek sem igazi kategorizálási hibák, mivel a modell felismerte, hogy melyik kategóriába tartozik az adott termék, csak az adott alcsoport helyett egy másik, szűkebb kategóriát határozott meg.

Voltak olyan esetek is, amelyeknél a termék neve alapján nem tudtam eldönteni, hogy az a kategória helyes-e, amit a modell prediktált, vagy amelyiket manuálisan írtak be a cég munkatársai. Ezek kerültek a „Terméknév alapján nem derül ki a pontos kategóriája” nevű csoportba, de a könnyebb megértés érdekében erre is hoztam két példát, amelyet a 6. táblázatban foglaltam össze.

6. táblázat: Példa a Terméknév alapján nem derül ki pontosan csoportra

Cikk neve	**Hibás kategória**	**Jó kategória**	**Alcsoport2**
LEO XKS-250P szivattyú	Felszíni szivattyú	Búvárszivattyú	Búvárszivattyú tisztavízre
Unicraft PHH 1001 raklapemelo	Akkus önjáró rak-lapemelő	Hidraulikus rak-lapemelő	Hidraulikus raklapemelő

Forrás: Sajátszerkesztés, 2024

[4] Ennek magyarázatát az adattisztítási fejezetben részleteztem

Az „Egyéb” kategóriába került minden olyan termék, amelyeknél ki tudtam következtetni, hogy mi lehetett a tanítás során a probléma, de elég egyedi problémák voltak ahhoz, hogy ne tudjam semelyik másik hibázási típusba besorolni. Ezt a csoportot a 7. táblázat szemlélteti:

7. táblázat: Egyéb kategóriába sorolt termékek

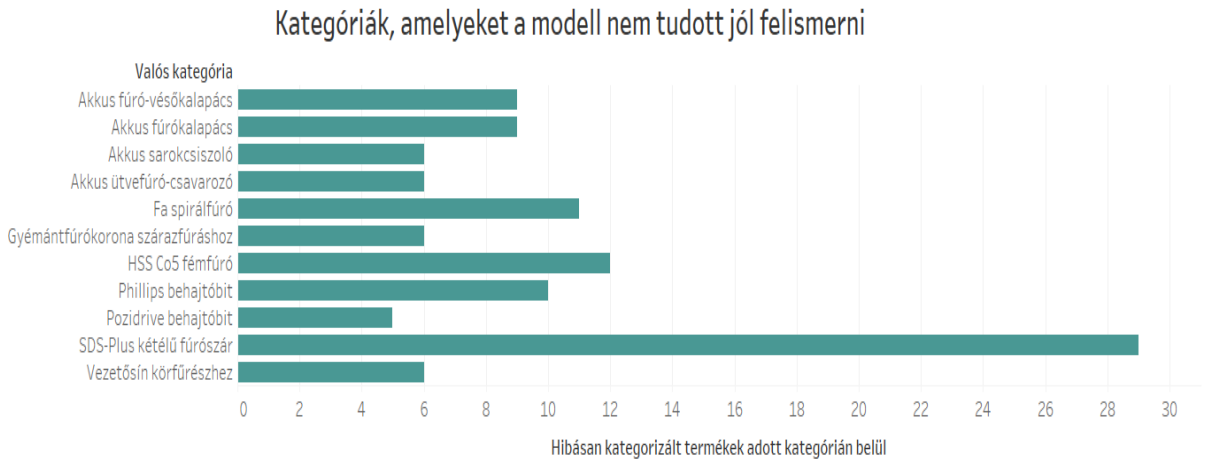
Cikk neve	**Hibás kategória**	**Jó kategória**	**Alcsoport2**
Bernardo MBS 350 DGA félautomatikus kettos-sarkalószalagfurész, 400 V	Elektromos félautomata betonkeverő	Szalagfűrész	Asztali fémipari szalagfűrész
Ztrust WH-201H fejpajzs külső védőplexi	Kézi hegesztő fejpajzs	Pajzsüveg	Hegesztő pajzs-üveg
Makita SK103PZ keresztlézer	Vonallézer	Pont- és vonal-lézer	Pont- és vonal-lézer

Forrás: Sajátszerkesztés, 2024

A táblázatban szerepelt „*Bernardo MBS 350 DGA félautomatikus kettos-sarkalószalagfurész, 400 V*” véleményem szerint azért kategorizált rosszul a modell, mert helyesírásilag nem volt megfelelően elnevezve a terméknév, és emiatt nem tudta értelmezni. A „*Ztrust WH-201H fejpajzs külső védőplexi*” terméknév esetén azt feltételeztem, hogy a fejpajzs szóra asszociált nagyon a modell, emiatt kategorizálta egy olyan csoportba, amelyben szintén szerepelt a fejpajzs szó. A „*Makita SK103PZ keresztlézer*” esetén viszont már érdekesebbnek bizonyult a kategoriázás. Ehhez hasonló terméknévvel még egy másik termék használt, amely a „*Flex ALC 3/1-basic keresztlézer*” volt. Mindkét terméknévben egyetlen magyar szó volt, hogy keresztlézer, amely alapján csoportosított maga a modell is. Az utóbbi termék valószínűleg benne volt a tanuló adathalmazban, mert az eredeti tábla szerint ennek a terméknek ezek voltak a kategóriái és mivel nagyon hasonló volt az elnevezésük, így a szerint osztályzott maga a modell is.

A „Ténylegesen rosszul kategorizált” csoportba kerültek azok a termékek, ahol a nevükből egyértelműen kiderült, hogy rosszul kategorizált a modell. Ezt szerettem volna alaposabban is elemezni, és egy oszlopdiagram segítségével összegezni azokat a alcsoportokat, amelyeknél kimagaslóan többször kategorizált rosszul a modell. Ehhez a

tanulmányaimból szerzett tudás alapján a Tableau alkalmazással gyorsan és egyszerűen létre tudtam hozni. A végeredményt pedig a 21. ábra szemlélteti.:



21. ábra: Ténylegesen rosszul kategorizáltak helyes kategóriái

Forrás: Sajátszerkesztés, 2024

Itt előbb leszűrtem a modell által hibásan prediktált termékek halmazát azokra az értékekre, amelyek ezen utolsó hibázási csoportba lettek besorolva, majd összegeztem, hogy egy adott kategórián belül hány darab terméknév fordul elő. Mindezt mindezt tovább szűrtem, hogy csak azokat mutassa, ahol a terméknevek száma meghaladja az 5 darabot. Itt látható, hogy különösen az SDS-Plus kétélű fúrószár esetén fordult elő a félre kategorizálás. Ezen esetben összesen 29 terméket nem sikerült megfelelően bekategorizálni. Itt az történhetett, hogy a tanító adathalmazban túl kevés olyan adat szerepelt, amely ebbe a kategóriába tartozott volna, emiatt nem tudta jól bekategorizálni a modell.

Ezek az értékelések azért jelentősek, mert ezek alapján lehet megérteni, hogy a modell, miként működik, mik voltak az esetleges hibák, hol lehet fejleszteni. Összességében elmondható, hogy a modell hatékonyan működik, mivel több mint 8 ezer terméknevet több mint 93%-os pontossággal tudott kategorizálni, így véleményem szerint megbízhatónak lehet tekinteni. Miután átnéztem a több mint 600 hibásan kategorizált termékek neveit, megállapítottam, hogy a vállalatnak pontosabb elnevezést kell adjon bizonyos termékek esetén ahhoz, hogy a modell még alacsonyabb hibaaránnyal dolgozzon.

Fenn állt az a lehetőség is, hogy ha biztosítom azt, hogy a teszt adatokban csak olyan termékek nevei szerepeljenek, amik több mint 5-ször szerepelnek az adathalmazban, így ezt is leteszteltem. Viszont a tanítási időt sem, de a hibaarányt sem tudtam csökkenteni. Ebben az esetben 1 tized ponttal tudtam növelni a találati helyességet, amely véleményem szerint egy jelentős javulás, ráadásul, ha a modellnek olyan új terméket kell besorolnia, amely kategóriája

nem szerepel ezen csökkentett tanítási adathalmazban, akkor bizonyosan rosszul fog prediktálni, hiszen nem is tud a megfelelő kategória létezéséről.

8. ÉRTÉKELÉS

Kutatásom legelején sok kérdés foglalkoztatott, már alapvetően a modell gyakorlati megvalósításával kapcsolatban is. Bár sok cég valószínűleg alkalmazza a BERT modellt kategorizálásra, még sincs róla igazán publikus adat. Ahol találtam valamennyit ott is inkább csak egy bináris gatív kategorizálási rendszert állítottak fel, nem pedig a nekem kellő, kettőnél jóval több kategóriát kezelni képes modellt. Az Amazon tudás cikkei voltak számomra a legnagyobb segítség, mivel ez volt az egyetlen olyan cég, amely nyilvánosan osztott meg egy olyan tudományos cikket, melyben leírják, hogy termékek kategorizálására használnak BERT modellt.

Ezek után, jött a megfelelő modell és környezet kiválasztása. Mivel a személyes számítógépem nem rendelkezett megfelelő mértékű erőforrásokkal, emiatt kellett keresnem egy alternatívát. A Google Colab rendszerét választottam végül, mivel ez volt az a környezet, ahol könnyen és gyorsan be tudtam tanítani a modellem. Az elején viszont abba a problémába ütköztem, hogy az ingyenes verziója nem biztosított elegendő erőforrást a számomra, így végül a Pro verziót használtam, hogy a tesztelési folyamatot még gyorsabban véghez tudjam vinni.

Kutatásaim során rájöttem, hogy nem kell egy teljesen új modellt létrehoznom, hanem elég egy már betanult modellt célomnak megfelelően finomhangolnom. Két modellt találtam, amelyek a feladatnak megfelelőnek tűntek, ám a futási eredményeik után végül csak egyet használtam. Az egyik a mBERT volt, a másik a HuBERT. Az előbbi, bár még mindig jobban teljesített, mint az emberi manuális kategorizálás, a magyar nyelven betanult BERT modellnek sikeresebbek lettek az eredményei, emiatt inkább ez utóbbit használtam a további teszteléseim során. Ezzel a modellel kevesebb mint 7%-os hibaaránnyal tudtam elvégezni a kategorizálási folyamatot, amely javulást jelent a manuális esetben jellemző 10%-hoz képest. Végül részletesen ellenőriztem, hogy milyen esetekben kategorizált félre a modell, a hibás működés megértése érdekében.

A finomhangolás során különösen figyelnem kellett a megfelelő paraméterezésre, hiszen, nem rendelkeztem korlátlan erőforrásokkal, illetve idővel sem. Át kellett gondolnom például a batch méret megadását, mivel, ha túl nagy méretet választottam volna, akkor a modellem betanítása nem tudott volna lefutni, a túl nagy erőforrás igény miatt. A másik paraméter, amire figyelnem kellett, a megfelelő epoch számok meghatározása, ám ezt egy korai megállással sikerült úgy beállítanom, hogy ne kelljen túl sokszor végigmenni a tanítási

adatokon, ahhoz, hogy meghatározzam a megfelelő számot. Így körülbelül 2 óra alatt sikerült finomhangolnom a modell

Nagy nehézséget okozott volna a teszteléseim során, ha minden alkalommal újra kellett volna tanítanom a modellt, hogy dolgozhassak vele, emiatt kerestem egy olyan megoldást, amivel el tudtam menteni a modellt a finomhangolási folyamat végén. Ezt sikerült is véghez vinnem, egy egyszerű mentési kóddal, amely tömörítette az adatokat, majd lementette a modellt a saját tárhelyemre, hogy a későbbiekben fel tudjam használni a már finomhangolt rendszert.

Végeredményben elmondható, hogy egy olyan modellt hoztam létre, amely képes helyettesíteni az emberi manuális kategorizálást, ezzel is segítve a vállalat munkafolyamatát. A jövőben lehetne még tovább tesztelni a modellt, hogy azoknál a csoportoknál, ahol nagyon magas volt a hibás kategorizálási arány, ott nagyobb figyelmet fordítson a termékek elnevezésére. Egy még komplexebb és hatékonyabb fejlesztési irány lehet, hogy a modell nem csak a terméknevek alapján prediktál, hanem az árukhoz csatolt képek feldolgozásával is segíti a kategorizálási folyamatot.

9. ÖSSZEFOGLALÁS

A kutatásom során egy mesterséges intelligencián alapuló kategorizáló modell fejlesztésére összpontosítottam, amely egy webáruház termékeinek automatikus kategorizálását végzi, pusztán a termékek nevéből kiindulva. Célom az volt, hogy a vállalat jelenlegi manuális folyamatainál gyorsabb és pontosabb megoldást biztosítsak, csökkentve ezzel a munkaerő-ráfordítást és a hibalehetőségeket. A projekt során a HuBERT modellt választottam, amely már egy magyar nyelvre optimalizált rendszer. Ezt pedig finomhangoltam a webáruház igényeihez igazodva. A modell kétirányú kontextus-elemzési képessége lehetővé tette, hogy pontosan felismerje a terméknevekhez tartozó kategóriákat, még összetettebb elnevezések esetén is.

Mivel a kategorizálás területén jelentős hatékonysági előrelépést vártam, nagy figyelmet fordítottam a modell finomhangolási folyamatára és az adathalmaz megfelelő előkészítésére. A webáruház biztosította több mint 45 ezer terméknév és kategória adatát, amelyek primer forrásként szolgáltak a modell tanításához. Az adathalmazban több mint 2000 kategória szerepelt, így a modell célja az volt, hogy ezeket egyértelműen azonosítani tudja a terméknevek alapján.

A projekt során kihívást jelentett, hogy a személyes laptopom számítási kapacitása nem volt elegendő a modell teljes körű betanításához. Ennek megoldására felhő alapú környezetet, a Google Colabot használtam, ahol elegendő számítási erőforrást tudtam biztosítani a modell

számára. A finomhangolási folyamat során olyan paraméterbeállításokat alkalmaztam, amelyek a modell pontosságát optimalizálták, és biztosítottam, hogy a tanító adathalmazban szereplő kategóriák mindegyike megfelelő súllyal képviseltesse magát. Ez különösen fontos volt, mivel a ritkább kategóriák felismerése nagyobb kihívást jelent a modell számára, és csak úgy érhető el nagyobb pontosság, ha ezekből is megfelelő mennyiségű adatot tartalmaz a tanító halmaz.

A kutatás eredményei túlszárnyalták a kezdeti elvárásaimat. A modell képes volt pusztán a terméknevek alapján kategorizálni az új termékeket, ezzel jelentősen csökkentve a manuális munka igényét és a hibák előfordulását. Az új rendszer a korábbi, manuális kategorizáláshoz képest gyorsabban és pontosabban működik, kevesebb mint 7%-os hibaaránnyal. Az eredményes működés igazolja a mesterséges intelligencián alapuló automatizált rendszerek hatékonyságát a vállalati szférában, különösen olyan területeken, ahol nagy mennyiségű adatot kell gyorsan és pontosan feldolgozni.

A modell tesztelése és az elvégzett elemzések során egy olyan fejlesztést javasolnék a cég számára, hogy a termékek nevének megadása során pontosabb, részletesebb leírásokat biztosítsanak, mivel ez tovább növelné a modell besorolási pontosságát. Ez különösen a ritkább kategóriák esetében bizonyulna hasznosnak, ahol kevesebb adat áll rendelkezésre a tanításhoz. Az eddigi tapasztalatok alapján, ha a tanító adatokban nagyobb arányban szerepelnének ezek a ritka kategóriák, a modell jobban felismerné ezeket is, így biztosítva, hogy a hibaarány még tovább csökkenjen.

Az eredményekkel ugyan elégedett vagyok, de úgy vélem, hogy további finomhangolással és az adathalmaz kibővítésével a modell még tovább optimalizálható. Emellett megfontolandónak tartom egy multitask modell alkalmazását is, amely nemcsak a terméknevek, hanem a termékek képei alapján is képes lenne kategorizálni, ezáltal vizuális alapú besorolási képességet biztosítva. Ez különösen hasznos lehet olyan esetekben, amikor a termék neve önmagában nem nyújt elegendő információt a pontos kategorizáláshoz.

Összességében a projekt igazolta, hogy a mesterséges intelligencia és a BERT modell alkalmazása jelentős előrelépést jelenthet a nagyvállalati kategorizálási folyamatokban. Az általam létrehozott modell nemcsak a munkafolyamatok hatékonyságát növeli, de pontosabb, automatizált kategorizálást tesz lehetővé, amely minimális emberi beavatkozást igényel. Az eredmények alapján egyértelmű, hogy az MI-alapú rendszerek alkalmazása nemcsak a mai követelményeknek felel meg, de folyamatos fejlesztési lehetőségeket is nyújt a jövőre nézve. Az automatizált kategorizálási rendszer bevezetése jelentős üzleti értéket képvisel, hiszen

csökkenti az erőforrásigényt, növeli a pontosságot és megbízhatóságot, így hozzájárulva a cég versenyképességének fenntartásához és növeléséhez.

IRODALOMJEGYZÉK

Russell, S., & Norvig, P., 2020. *Artificial Intelligence: A Modern Approach* (4th ed.).

Goodfellow, I., & Bengio, Y., & Courville, A., 2016. *Deep Learning*.

Jurafsky, D., & Martin, J. H., 2020. *Speech and Language Processing* (3rd ed.).

Devlin, J., & Ming-Wei C., & Lee, K., & Toutanova, K., 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [online] Elérhető: https://arxiv.org/abs/1810.04805 [Hozzáférés dátuma: 2024.09.15]

Vaswani, A., & Shazeer, N., & Parmar, N., & Uszkoreit, J., & Jones, L., & Gomez, A. N., & Kaiser, L., & Polosukhin, I, 2017. *Attention is all you need*. [online] Elérhető: https://arxiv.org/abs/1706.03762 [Hozzáférés dáutma: 2024.09. 11]

Kajal., 2023. *Step-by-step BERT implementation guide*. Analytics Vidhya. [online] Elérhető: https://www.analyticsvidhya.com/blog/2023/06/step-by-step-bert-implementation-guide/ [Hozzáférés dátuma: 2024. 09. 11]

Muller, B., 2023. *BERT 101: State of the art NLP model explained*.[online] Elérhető: https://huggingface.co/blog/bert-101#3-bert-model-size--architecture [Hozzáférés dátuma: 2024.09.20]

Luna, J., C., 2023 *What is BERT? An Intro to BERT Models* [online] Elérhető: https://www.datacamp.com/blog/what-is-bert-an-intro-to-bert-models [Hozzáférés dátuma: 2024.09.21]

Jeeva, J., 2023. *Pre-training the BERT model* [online] Elérhető: https://www.scaler.com/topics/nlp/pre-training-bert/ [Hozzáférés dátuma: 2024.09.21]

High Plains Computing, 2023. *An intuitive guide to low-rank adaptation (LoRA), quantization, and fine-tuning an LLM*. [online] Elérhető: https://highplains.io/an-intuitive-guide-to-low-rank-adaptation-lora-quantization-and-fine-tuning-an-llm/ [Hozzáférés dátuma: 2024.09.12]

Innodata, 2023. *Artificial general intelligence vs. generative AI: Which is the future?* [online] Elérhető: https://innodata.com/artificial-general-intelligence-vs-generative-ai-which-is-the-future/ [Hozzáférés dátuma: 2024.09.11]

Gyires-Tóth, B., 2019. *A mélytanulás múltja, jelene és jövője*

Nelson, D., 2024. *Mi az NLP (természetes nyelvi feldolgozás?)* [online] Elérhető: https://www.unite.ai/hu/mi-a-term%C3%A9szetes-nyelvi-feldolgoz%C3%A1s/ [Hozzáférés dátuma: 2024. 09. 11]

IBM, 2023. *AI vs. machine learning vs. deep learning vs. neural networks* [online] Elérhető: https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks [Hozzáférés dátuma: 2024.09.11]

Aayush, M., 2023. *NLP Rise Transformer Models | A T5, a BERT és a GPT átfogó elemzése* [online] Elérhető: https://www.unite.ai/hu/nlp-rise-with-transformer-models-a-comprehensive-analysis-of-t5-bert-and-gpt/ [Hozzáférés dátuma: 2024. 09. 12]

Ba, J. L., & Kiros, J. R., & Hinton, G. E., 2016. *Layer normalization.* [online] Elérhető: https://doi.org/10.48550/arXiv.1607.06450 [Hozzáférés dátuma: 2024.09.21]

Liu, Y., & Ott, M., & Goyal, N., & Du, J., & Joshi, M., & Chen, D., & Levy, O., & Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach*. [online] Elérhető: https://arxiv.org/abs/1907.11692 [Hozzáférés dátuma: 2024.10.01]

Mosbach, M., & Andriushchenko, M., & Klakow, D., 2021. *On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines*. [online] Elérhető: https://arxiv.org/abs/2006.04884 [Hozzáférés dátuma: 2024.10.01]

Srivastava, N., & Hinton, G., & Krizhevsky, A., & Sutskever, I., & Salakhutdinov, R., 2014. *A simple way to prevent neural networks from overfitting* [online] Elérhető: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf [Hozzáférés dátuma: 2024.10.01]

Wesley, F. M., & Carmignani, A., & Bortoli, G., & Maretti, L., & Luz, D., & Guzman, D. C. F., & Henriques, M. J., & Neto, F. L., 2024. *Multi-level Product Category Prediction through Text Classification* [online] Elérhető: https://arxiv.org/abs/2403.01638 [Hozzáférés dátuma: 2024. 10. 02]

Zahera, H. M., & Sherif, M. A., 2020. *ProBERT: Product Data Classification with Fine-tuning BERT Model* [online] Elérhető: https://www.semanticscholar.org/paper/ProBERT%3A-Product-Data-Classification-with-BERT-Zahera-Sherif/9aec895bb352037ad1248603cc2d90c76582bab2 [Hozzáférés dátuma: 2024.10.07]

Jagrič, T., & Herman, A., 2024. *AI Model for Industry Classification Based on Website Data* [online] Elérhető: https://www.mdpi.com/2078-2489/15/2/89 [Hozzáférés: 2024.10.07]

Nemeskey, D. M., 2020. *Natural Language Processing Methods for Language Modeling* [egyetemi disszertáció] Budapest: Eötvös Loránd Egyetem, Elérhető: https://hlt.bme.hu/media/pdf/nemeskey_thesis.pdf [Hozzáférés dátuma: 2024. 10. 02]

Dr. Kovács, E., 2024. Mesterséges Intelligencia kurzus

Kuknyó, D., 2020. *MESTERSÉGES INTELLIGENCIÁVAL TÁMOGATOTT DINAMIKUS ÁRAZÓRENDSZER KIALAKÍTÁSA AZ ONLINE KERESKEDELEM FEJLESZTÉSÉRE*

[szakdolgozat] Alapképzés. Budapest: Budapesti Gazdasági Egyetem, Pénzügyi és Számviteli Kar. URL [Hozzáférés dátuma: 2024. 06.12]

Hastie, T., & Tibshirani, R., & Friedman, J., 2009: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition

Alwis, M., 2020. *Google's BERT Update: The NLP Framework Transforming Search Engines* [online] Elérhető: https://www.adaptworldwide.com/insights/2020/googles-bert-update [Hozzáférés dátuma: 2024. 10. 10.]

Koroteev, M. V., 2021. *BERT: A Review of Applications in Natural Language Processing and Understanding* [online] Elérhető: https://doi.org/10.48550/arXiv.2103.11943 [Hozzáférés dátuma: 2024. 10.09]

Amazon Machine Learning, 2016. *Model Fit: Underfitting vs. Overfitting* [online] Elérhető: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html [Hozzáférés dátuma: 2024. 10. 10]

Gyetvai, N., 2023. *Termékek kategorizálása? Megmutatjuk a legfontosabb alapelveket, hogy hogyan kategorizáld helyesen a termékeidet egy webáruházban!* [online] Elérhető: https://webstartconsulting.hu/termekek-kategorizalasa-megmutatjuk-a-legfontosabb-alapelveket-hogy-hogyan-kategorizald-helyesen-a-termekeidet-egy-webaruhazban/ [Hozzáférés dátuma: 2024. 10. 21]

Qiu, Y., & Jin, Y., 2024. *ChatGPT and finetuned BERT: A comparative study for developing intelligent design support systems* [online] Elérhető: https://doi.org/10.1016/j.iswa.2023.200308 [Hozzáférés dátuma: 2024. 10. 12]

Rogers, A., & Kovaleva, O., & Rumshisky, A. 2021. *A Primer in BERTology: What We Know About How BERT Works* [online] Elérhető: https://doi.org/10.1162/tacl_a_00349 [Hozzáférés dátuma: 2024. 10. 19]

EITCA Akadémia, 2024. *Milyen példák vannak a félig felügyelt tanulásra?* [online] Elérhető: https://hu.eitca.org/mesters%C3%A9ges-intelligencia/eitc-ai-gcml-google-felh%C5%91g%C3%A9pes-tanul%C3%A1s/bevezet%C3%A9s-a/mi-a-g%C3%A9pi-tanul%C3%A1s/milyen-p%C3%A9ld%C3%A1k-vannak-a-f%C3%A9lig-fel%C3%BCgyelt-tanul%C3%A1sra/ [Hozzáférés dátuma: 2024. 10. 15]

MELLÉKLETEK

I. Füri_Erika_Rebeka_PSZK_BGE_TDK_Szerzői hozzájárulás nyilatkozat 2024

II. Füri_Erika_Rebeka_PSZK

III. Adatoktisztitasa (adattisztítási folyamat – python kódsor)

IV. HuBert_Modell (HuBERT modell tanítása – python kódsor)

V. HU_BERT2 (HuBERT modell kisebb számú alcsoportok kivevéssel teszt – python kódsor)