huBERT alapú sziámi neurális háló architektúrák elemzése ügyfélszolgálati emailek klasszifikációjára

Vándor Péter[1] és Csáki Csaba[1]

[1] Budapesti Corvinus Egyetem, Fővám tér 8,
H-1093 Budapest, Magyarország
peter.vandor@stud.uni-curvinus.hu
csaki.csaba@uni-curvinus.hu

Kivonat: A digitális gazdaságban megnőtt az ügyfélszolgálatok szerepe és az ügyfelek nem-közvetlen kommunikációs csatornák (pl. email) esetén is gyors választ és hatékony megoldást várnak. Az ügyfélszolgálatoknak ezt úgy kell elérniük, hogy közben a szöveges (email, közösségi média stb.) üzenetek száma rohamosan növekszik. A szöveges adathalmazok természetes nyelvi feldolgozásának egyik új technikája a BERT, melynek huBERT néven magyar változata is elérhető. Az itt bemutatott kutatás célja kettős: egyrészt egy valós ügyfélszolgálati probléma (nagy mennyiségű emailek automatikus kategorizálásának) mesterséges intelligencia alapú informatikai támogatása a hatékonyabb feldolgozás érdekében; másrészt a felhasznált BERT-alapú MI architektúrák szisztematikus vizsgálata tapasztalat szerzés céljából. A Python nyelven megírt és huBERT modulokat is használó sziámi architektúrára épülő 3 illetve 10 kategóriát felismerő, ügyfél emaileket feldolgozó megoldás paramétereinek változtatásával növelhető volt a pontosság és az alkalmazott legösszetettebb struktúrával a kezdeti alig 70%-os helyett közel 95%-os teljesítmény volt elérhető.

1 Ügyfélszolgálati kihívások és kapcsolódó NLP technikák

A hálózatos digitális gazdaság egyik hozadéka, hogy megnőtt az ügyfélszolgálatok szerepe mind az ügyfélelégedettség mind a színvonalasabb szolgáltatás nyújtás vonatkozásában (Cassandra és mtsai., 2019). A változó feladatokkal és újabb technológiákkal és kommunikációs csatornákkal (email, web űrlapok, közösségi média, videó hívások stb.) bővülő felelősségi kör komoly terhelést helyezett erre a vállalati funkcióra (Zendesk, 2020). Az új csatornák, és különösen az email általános elérhetősége miatt megnövekedett a beérkező üzenetek, kérések, panaszok, és kérdések száma. Ráadásul egyes 'kampány' időszakokban különösen kiugró lehet a kezelendő megkeresések száma. Ezeket a beérkező üzeneteket sokszor rövid határidővel kellene rendezni, mivel az ügyfelek nem közvetlen kommunikációs technikák (pl. email) esetén is gyors választ és hatékony megoldást várnak. A megnövekedett ügyfélszolgálati terhelés és elvárások együttes hatásának kivédésére egyre nagyobb hangsúlyt kapnak a mesterséges intelligencia (MI) alapú megoldások, azon belül is a természetes nyelv feldolgozás (angol rövidítésével NLP) valamilyen formája vagy technikája (Borg és mtsai., 2021) – így az ügyfélszolgálatok digitalizációjának része lett az NLP (Indurkhya és Damerau, 2010).

A mesterséges intelligencia technikák egyik ígérete, hogy komplex, nagy adathalmazokban is képesek az embernél hatékonyabban mintákat felismerni (Ferrario és Loi, 2022). A legelterjedtebb (legkézenfekvőbb) lehetőség az ún. chatbotok alkalmazása (Mnasri, 2019; Nguyen, 2019), mely lehet szabály alapú vagy nyílt szöveges megoldás. Ezen túlmenően felmerülhet telefonos beszélgetések értelmezése, közösségi média bejegyzések elemzése (pl. sentiment vagy topic analysis, vagy legalább a bejövő üzenetek tartalom alapján történő kategorizálása, előkészítése, vagy akár megválaszolása (Hardalov és mtsai., 2018; Hardalov és mtsai., 2019; Borg és mtsai., 2021). Ezek azonban sokszor nem elegendők önmagukban (Hardalov és mtsai., 2018; Saberi és mtsai, 2017). Ezért e technikák mellett egyre nagyobb figyelmet kapnak az összetetteb szöveg elemző technológiák, mint pl. a GPT-3 vagy újabban a BERT modellek (Devlin és mtsai.; 2018; Finardi és mtsai., 2021). Ezek lényege, hogy mélytanulásos megoldást alkalmaznak ún. 'transformer' hálózati felépítéssel és a 'figyelem'-re (attention) épülő modell alapelemekkel. A betanítás során igen nagyszámú (akár tízmilliárdos nagyságrendű) bemeneti paramétert alkalmazva, autoregresszív módon építik fel egy adott nyelv modelljét úgy, hogy a nyelvet reprezentáló adatok (a corpus) több tíz gigabájtnyi méretű is lehet (pl. rengeteg online weboldal, vagy hatalmas mennyiségű irodalmi mű), a céltól függően. Ebből adódik, hogy a modell kialakítása gépi erőforrás és idő igényes. Az elkészült modell azonban több célra használható (pl. szövegből hiányzó szavak beillesztésére), jellemzően úgy, hogy a modell köré összetettebb architektúrát építenek további, egyszerűbb modellekből. Előnyük, hogy egyrészt hosszabb (bár tokenizált) szövegeket is kezelni tudnak, másrészt ezt úgy teszik, hogy információt képesek kinyerni az adott szöveg vagy szöveg részek kontextusára vonatkozóan is, és ennek megfelelően javasolnak megoldást (a feladattól vagy kérdéstől függően). Egy ilyen modellt adott nyelvre elő kell készíteni, ami hatalmas erőfeszítés – viszont huBERT néven már magyar változat is elérhető (Nemeskey, 2021). Ugyanakkor a BERT modelleket más neurális háló megoldásokkal is lehet kombinálni a feladattól függően (Lu és mtsai.; 2020; Yang és Cui, 2021).

Jelen tanulmányban bemutatott kutatás a SZTAKI által kidolgozott huBERT alapmodellre (Nemeskey, 2021) építve vizsgált különböző architektúrákat és méreteket egy konkrét cég ügyfélszolgálatára beérkező (különböző hosszúságú) email üzenetek klasszifikációs előkészítése céljából. A munka számos lépésben igyekezett egyre jobb teljesítményű architektúrát kialakítani.

2 Az eset háttere és adatok

A kutatás alapja egy email kategorizáló megoldás kidolgozása egy óvodai, iskolai étkeztetés adminisztrációját automatizáló cég (E-Menza, e-menza.hu) számára (Netkir, 2021). Regisztrált szülők a cég jelszóval védett webes felületén meghatározott beosztás szerint adhatják le a menzai étkezések megrendelését. A rendeléssel kapcsolatos kommunikáció (allergia, lemondás, stb.) illetve technikai jellegű kérések (pl. elfelejtett jelszó vagy egyéb bejelentkezési problémák) intézése alapvetően emailen keresztül történik. A rendelés időszaki jellege miatt az érkező e-mailek számában napi szinten igen kiugró értékek is lehetnek. Az is előfordulhat, hogy egy emailben több kérés vagy bejelentés is szerepel. Az e-maileket azok tartalmától függően kell elosztani az

ügyfélszolgálati referensek között, akik aztán döntenek más dolgozók (pl. pénzügyesek vagy IT technikusok bevonásáról).

A központi postafiókba összesen több mint 600 000 e-mail érkezett, azonban ezek többsége reguláris kifejezésekkel kezelhető volt (például 365 084 db rendelés visszaigazolás, 146 604 kézbesíthetetlen). A 10 000 körüli maradék halmazból csak az ügyfél indító levelét vettük figyelembe, a kommunikáció többi részét kiszűrtük. Így összesen 657 e-mail maradt. A postafiókba beérkező leveleket kézzel osztályozták a cég ügyfélszolgálati munkatársai és 10 kategóriát alakítottak ki a leggyakrabban előforduló problémakörökből: allergia, bankkártya, cafeteria, iskolaváltás, jelszó, kedvezmény, már beküldött kedvezmény, lemondás, regisztrációs kód és utalás. A betanításhoz mindegyik osztályba nagyjából 50 db e-mail került, legfeljebb 3-4 db email eltéréssel. Így egy kiegyensúlyozott bemeneti adathalmaz keletkezett, ami összesen 508 db e-mailt tartalmazott. A teljes maradék halmazból kiszűrtük az egyedi, sokféle kategóriába besorolható elemeket és így a 10 kategória eloszlása ott is hasonlóan egyenletes képet mutatott. Egyik kategóriában sem volt szignifikánsan több elem mint a többiben.

A feladat tehát elsőre egyszerű besorolásnak tűnhet (pl. a tárgy alapján). Ugyanakkor hamar kiderült, hogy az e-mailek címe nem mindig egyértelmű, az egyes ügyfelek szóhasználata széles skálán mozog. Sőt, a felhasználók több esetben a rendszertől kapott figyelmeztető levelekre válaszolva írták meg problémáikat egy teljesen más tárgykörben. Az e-mailek szövege igen eltérő hosszúságú is lehet. A fentiekből adódott, amit néhány egyszerű, hagyományos klasszifikációs gépi tanuló technika alkalmazásával végrehajtott teszt igazolt is (1. táblázat), hogy összetettebb NLP megoldás alkalmazása lehet célravezető. Az előzetes teszt eredményeként az e-mail tárgyát végül ki is kellett hagyni az elemzésből. A rendelkezésre álló kevés adat miatt nem tudtuk finomhangolni a BERT modellt, a teszteléshez pedig csak két halmazra bontottuk az adatokat és ezért olyan módszert is kerestünk, mely képes néhány példa alapján tanulni.

1. Táblázat: Viszonyítási alapul szolgáló gépi tanuló algoritmusok eredményei

Modell	Random Forest	Naive Bayes	Logistic Regression	SVM
Validációs pontosság	71,64%	76,36%	77,94%	83,65%

3 Alkalmazott módszerek

A megoldás kidolgozásához a BERT magyar változatára, a huBERT-re esett a választás mivel az ügyfélszolgálatokra beérkező e-mailek a jellemzően egyszerű hétköznapi nyelvezetet használnak. Az új modellek optimalizálása során a huBERT-hez tartozó súlyok változtatása le lett tiltva. Emellett olyan megoldásra volt még szükség, amely alkalmas arra, hogy már néhány példából is tanuljon és képes legyen azonnal osztályozni a beérkező emaileket. Így esett a választás a sziámi hálózat megoldásra, melyet az ún. 'egy-lövéses-tanulás' (One Shot Learning – OSL) megközelítés alatt szoktak említeni. A gyorsabb tanulást úgy éri el, hogy az eredeti multiklasszifikációs problémát áttranszformálja párok összehasonlítására úgy, hogy az eredeti feladatot megoldó már betanított neurális hálózatból indul ki és azt a modellt duplikálva beágyazza egy

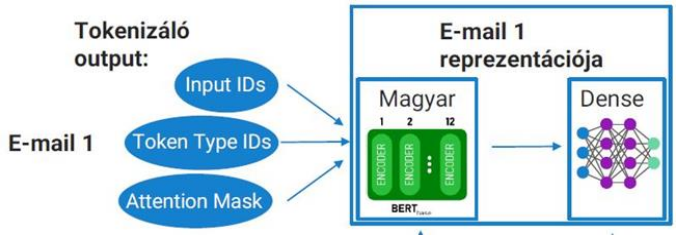
összehasonlító hálózati architektúrába (lásd lent). Ennek eredményeként a beágyazott hálózat által kialakított reprezentáció oly módon változik, hogy az egyes osztályok egymástól eltérő tulajdonságait próbálja detektálni és minél jobban elkülöníteni.

Programozásra a Python programnyelvet használtuk skálázhatósága, teljesítménye és a vállalati folyamatokba való integrálhatósága okán. A fejlesztés a Jupyter Notebook (v. 5.0), illetve az Anaconda (v. 3.8) környezetek segítségével történt. Általában az UTF-8 kódolás volt a jellemző. Egy e-mail lehet egy részből álló egyszerű szöveg (text/plain) vagy több részből álló (multipart). A HTML részeket kiszűrtük és egyszerű szöveggé konvertáltuk. Dekódoltuk a feladó, tárgy és törzsszöveg tartalmát. A törzs esetében megvizsgáltuk a részeket és több rész esetén megkerestük az első egyszerű szövegrészt, majd betöltöttük egy adattáblába. A tisztított törzsszöveget adtuk át a nyelvi modellnek. A BERT kétféle kimenettel dolgozik: ezek a teljes szekvenciát egyetlen 768 hosszú vektorral reprezentáló pooled output és az előzővel megegyező dimenziószámú vektorokat tokenenként legeneráló sequenced output.

A fenti környezetben a már előzetes tesztek után négy modell kiépítése történt meg, melyek mindegyikén több paraméter értékét is megvizsgálta a kutatás. A már betanított és működő de átlagos eredményt elérő osztályozót kellett első lépésben felépíteni, melyet később a sziámi beágyazott hálózatává alakítottunk. Az első kimenet kipróbálása után áttértünk a szekvenciális kimenet felhasználására és további rejtett rétegek hozzáadásával javítottunk az eredményen. Először magának az alapmodellnek a kialakításánál több lépésben került kiválasztásra egy LSTM-mel bővített megoldás, majd a sziámi architektúra alkalmazására került sor, a kész alapmodellre építve. A sziámi struktúrában egyrészt a belső hálózatok méretét (neuron számát) lehetett változtatni, másrészt a speciálisan kialakított batchek elemszámát, valamint a tanulási rátát optimalizálni. A kimeneti (osztályozásnál, illetve összehasonlításnál két aktivációs függvény vizsgálatára került sor: a két alapmodellnél softmax, a one-shot rendszernél pedig szigmoid.

3.1 huBERT + sűrű (Dense) multiklasszifikációs osztályozó

Az első modell, amely már huBERT megoldást használt (a tesztelés egyszerű gépi tanulási technikái helyett) egy bemeneti tokenizálóból és a huBERT utáni egyetlen sűrű (Dense) neurális háló rétegből állt (lásd 1. ábra).



1. ábra: huBERT + sűrű (Dense) multiklasszifikációs osztályozó

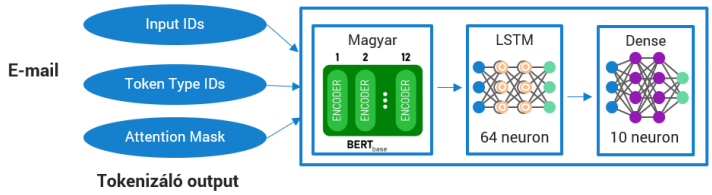
Az emailek feldolgozása során 64 tokenben maximalizáltuk a szekvencia hosszát. Az ennél rövidebb emaileket 0-kal kiegészítettük (padding) a maximális hosszig és attention mask jelezte, hogy ezeket a részeket ki kell hagyni a jelentés kialakítása szempontjából. Ezzel sikerült kiküszöbölni, hogy az E-Menza által küldött figyelmeztető

email is bekerüljön az értelmezett szövegbe. Az adattáblánk email törzsszöveg mezője alapján előre elkészítettük a BERT használatához szükséges 3 vektort: a bemeneti azonosítókat, típus azonosítókat és a figyelem maszkot. A huBERT szekvenciális kimenetét kifeszítés után egy 64 neuronból álló sűrű (Dense) réteghez kötöttük, amelynek rendeltetése egy 64 dimenziós reprezentáció kialakítása volt a huBERT 64 token x 768 vektor = 49 152 elemű kimenetéből. Ezzel a leszűkítéssel a neurális háló a legfontosabb jellemzőket kiemelve megfelelő, tömör reprezentációt alakít ki, amelyet már könnyebb osztályozni. Végül egy, a 10 kategóriának megfelelően 10 neuronból álló sűrű (Dense) réteg következik softmax aktivációs függvénnyel, amely egy valószínűségi eloszlást ad vissza. Az eloszlás legnagyobb értéke a megjósolt kategóriának megfelelő neuron aktivitásában mutatkozik meg.

A tokenizált bemeneteket betanító halmazra és validációs halmazra osztottuk. Fontos, hogy a validációs halmazba kiválasztott 10% megtartsa ugyanazt az eloszlást az osztályok közt, ami a teljes adathalmazra jellemző. 457 elem a betanító, 51 elem pedig a validációs halmazba került. A címkék szerinti eloszlást (stratify) megtartva a minimális elemszámbeli eltéréseket is sikeresen kezeltük. Az eloszlás megtartásától függetlenül a választás véletlenszerű, így a későbbi ellenőrzés, visszatöltés céljából a tanító és validációs halmazokat lementettük. Ez biztosítja, hogy a modell pontosan ugyanazt a tanító és teszt halmazt alkalmazza.

A modell paraméterezésekor az Adam optimalizálót adtuk meg, amely a momentum és az adaptív optimalizálás elvén működik. Az alapértelmezett tanulási rátát (0.001) nem módosítottuk. A loss értékét csak a pontosság mérésével egészítettük ki, a loss csökkentéséhez a multiklasszifikációs esetekben szokásos categorical_crossentropy függvényt adtuk meg. A batch méretét az adatok nagyságrendjéhez illeszkedően 64 hosszúságúra választottuk és az epochok számát többszöri futtatás után 20-ra állítottuk, mert a betanítási és validációs loss gyakorlatilag 15 epoch után ellaposodott. A modell összesen 113 764 554 paraméterrel rendelkezik, amely elsősorban a rétegek közti kapcsolatok súlyát jelenti. Mivel a huBERT rétegre letiltottuk a betanítást, ezért a modellnek csak 3 146 442 paramétere változtatható. A betanítás legvégén elmentjük a modell súlyait, hogy a későbbiekben bármikor visszatölthessük.

3.2 huBERT + LSTM + sűrű (Dense) multiklasszifikációs osztályozó



2. ábra: huBERT + LSTM + sűrű (Dense) modell felépítése

Az LSTM egy speciális RNN, ami a nagyobb távolságra elhelyezkedő, időben hosszú távú kontextust is képes elsajátítani. Különösen az információ hosszú távú megjegyzésére készítették őket. Egy szövegalapú NLP feladatban ez kritikus, hiszen akár több mondattal az aktuális előtt is lehet a szükséges információ. Ezért a jelentéstartalom

reprezentáció kiszámítására a sűrű hálózat helyett ezt alkalmaztuk. Az LSTM réteg 64 neuront tartalmazott és ezt követte egy 10 neuronból álló kimeneti Dense réteg a klaszszifikáció kiszámítása céljából (2. ábra). A modell összesen 110 832 010 paraméterrel rendelkezik, amelyből 213 898 paraméter tanítható.

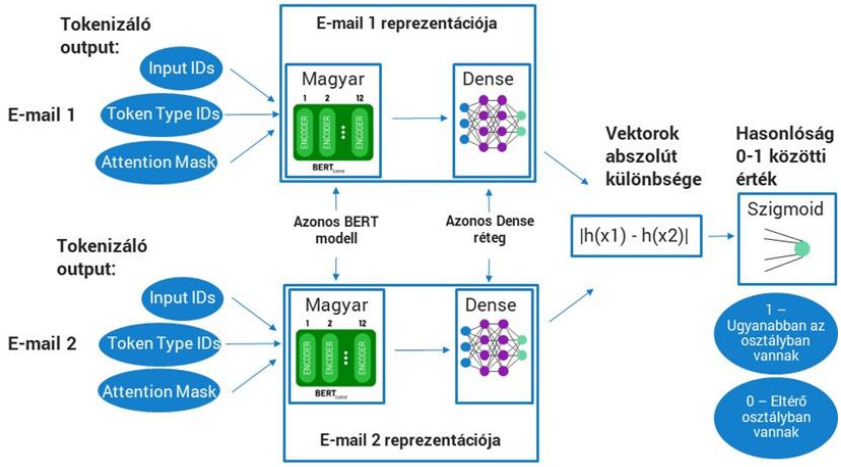
3.3 OSL sziámi hálózat két bemenettel és huBERT + sűrű (Dense) osztályozóval

Az előző architektúrák a multiklasszifikációs osztályozási feladatra jelentenek megoldást. Az egyes emailek összehasonlításán alapuló sziámi hálózatban a fenti architektúrákat beágyazott modellként használtuk fel. Ebben a modellben a huBERT + sűrű (Dense) hálózatot illesztettük be. A funkciója olyan reprezentáció kialakítása, amely az egy osztályba tartozó emaileket közelíti egymáshoz, míg a külön kategóriába tartozókat távolítja egymástól. A 2 bemenetre összeállított 16 hosszú reprezentációs vektorokat a dimenziónkénti abszolút különbségük kiszámításával a végső rétegben egy bináris döntéshez használjuk fel: a 2 vizsgált email ugyanabba vagy eltérő osztályba sorolható-e.

A huBERT-nek továbbított 3, tokenizált adatokat tartalmazó vektort kétszer tároljuk a bal, illetve jobb bemenetben egy tömbbé konkatenált formátumban. A huBERT modell után egy reprezentációt szűkítő 16 neuronból álló sűrű réteg kerül. Ezután a megduplázott hálózat kimenetét egy speciális Keras Lambda réteg kapja meg. A szokásos matematikai függvényeket nem szabad használni, hanem a Keras Backend könyvtára segítségével kell megadni a számításokat, mert ellenkező esetben megakadályozzuk a hálózat tanulását (csak ugyanazon loss értékek körül fog ugrálni a betanítás során). A Lambda réteg a párba állított reprezentációs vektorok dimenziónkénti abszolút különbségét számítja ki. A 16 dimenzió különbségét végül az egyetlen neuronból álló sűrű réteg kapja meg, ahol a szigmoid aktivációs függvény kiszámítja a hasonlósági értéket, ami 0 és 1 közötti érték. Ez a hasonlósági érték: 1, ha azonos osztályban vannak és 0, ha eltérő osztályba tartoznak az e-mailek. A 3. ábrán látható a teljes hálózat felépítése.

A súlyokat és az eltolást (bias) a belső sűrű rétegnél speciálisan inicializáltuk. A súlyoknál 0 középpontú, 10^{-2} varianciájú normális eloszlással dolgoztunk. Az eltolás esetén pedig egy 0,5 átlagú, előzővel azonos varianciájú normális eloszlást használtunk. A véletlenszerű inicializálással szemben ez a módszer biztosítja, hogy a betanítás indulásakor a kezdeti loss értékek sokkal jobbak legyenek és ennélfogva gyorsabban haladjon a hálózat.

A párosítással létrejött összes kombináción futtatni a modellt nagyságrenddel nagyobb számú iterálást jelentett volna, ezért a batchek saját összeállításával tartottuk kordában a lefutási időt. A 64-es batch méret helyett 16 email párt tartalmazó kötegelést alkalmaztunk, amelyen belül egyenlő arányban mutattunk véletlenszerűen kiválasztott pozitív (egyező kategória) és negatív (eltérő kategória) példákat a hálózatnak. Ha nem lenne kiegyensúlyozott a betanító halmaz, akkor ezzel a módszerrel biztosítani tudjuk az egyenlő arányokat is. Vegyük észre, hogy N elemű halmaz esetén a sziámi módszer N^2 nagyságrendű input teret alakít ki és ezért működik kiválóan kis N-ekre is. Például 10 email esetén (10x9)/2 pozitív párt tudunk összeállítani, ha a felcseréléseket nem számoljuk. A módszer másik következménye, hogy a betanítás nem epoch alapon zajlott, hanem a train_on_batch eljárás felhasználásával a saját párosításokkal. Ezért az iterációk számát a 20 epoch helyett 2000 batch-re kellett növelni.



3. ábra: OSL hálózat huBERT + sűrű (Dense) beágyazott modellel

Az Adam optimalizálónál csökkentettük az alapértelmezett tanulási rátát 6 x 10^{-5}-re, amely elegendőnek bizonyult a konvergencia eléréséhez. A modell nem módosítható paramétereinek száma 110 618 112, amely elsősorban a BERT modell súlyait jelenti. A 786 465 optimalizálható súlyt a BERT és az utána kötött 16 neuronból álló sűrű réteg kapcsolatainak a száma (49 152 x 16 = 786 432) plusz a sűrű és Lambda réteg közötti kapcsolatok (16), végezetül a Lambda és a végső 1 neuronnal rendelkező sűrű réteg összekötése (16 súly + 1 eltolás) biztosítja.

Az eredeti multiklasszifikációs probléma megoldása érdekében generáltunk egy minden osztályt pontosan egyszer magába foglaló támogató halmazt véletlenszerűen kiválasztott 1-1 db e-maillel. A támogató halmaz minden elemét összehasonlítottuk az OSL segítségével a kategorizálásra váró e-maillel. A legnagyobb hasonlósági érték alapján döntöttük el, hogy az adott e-mail melyik osztályba tartozik.

3.4 OSL sziámi hálózat 2 bementtel és huBERT + LSTM + Dense osztályozóval

A legtöbb paraméterben és működésében megegyezik az előző sziámi hálózattal, kivéve a beágyazott modellt, amely bővült egy LSTM réteggel, felhasználva a már letesztelt 2. multiklasszifikációs modellt. A gyorsabb betanítás érdekében dinamikus tanulási rátát vezettünk be. A betanítás elején jobb, ha nagyobb a ráta, mert nagyobbakat lépünk a gradiens irányába, míg a végén a minimum megtalálása érdekében, már egyre kisebb értékek alkalmazása a célravezető. A kezdeti 0,006 az összes iteráció legvégére 0,006 x 0,1(2000/2000)= 0,0006 értékre csökken. A fix tanulási ráta esetén majdnem 20.000 iteráción keresztül kellett tanítani a hálózatot a megfelelő eredmény eléréséhez.

4 Eredmények és azok diszkussziója

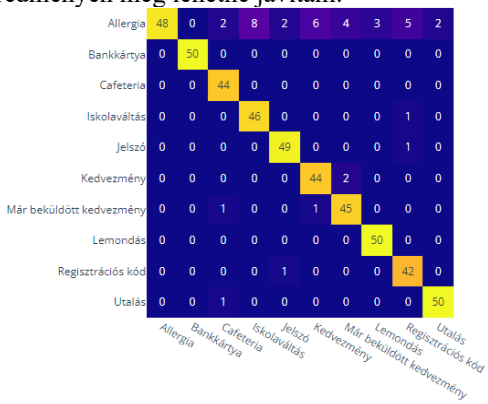
4.1 huBERT + sűrű (Dense) multiklasszifikációs osztályozó

A betanítási loss értéke a kezdeti 4 feletti értékről 0,008-0,007 szintekre csökkent, a pontosság 99,78%-on állt meg. De azért fontos az ellenőrzés a teszt halmazon az algoritmus által még nem látott emailekkel, hogy kiderüljön megfelelő általános szabályokat sajátított el a rendszer vagy sem. A validációs loss 1,41, míg a pontosság 68,6% körül alakult. A rendszer túlillesztette a betanító halmazt. A sűrű rejtett réteg esetén várható volt ez az eredmény, mivel ez a fajta réteg teljesen összekötött az előző réteg minden kimenetével és az adatok lineáris kombinációját képes csak előállítani, ami gyakorlatilag a klasszikus kulcsszó kiválasztási módszerekhez hasonlítható.

A teljes 508 számosságú halmaz tanulás utáni reprezentációs vektorait kinyertük. A TruncatedSVD és PCA függvények kétféle eljárással a 64 dimenziós vektorteret 3 dimenzióra redukálják. Az osztályok elkülönítése csak minimálisan sikerült, a legtöbb kategória feltűnően erős átfedésben volt.

4.2 huBERT + LSTM + sűrű (Dense) multiklasszifikációs osztályozó

A betanítási loss értéke a kezdeti 2 feletti értékről 0,36-ra csökkent, a pontosság 94%-on állt meg a betanító halmazon, de a validációs loss 0,76, míg a pontosság 80% körül alakult. A különbség csökkent a két görbe között és láthatóan megszűnt a túlillesztés problémája, de az eredményen még lehetne javítani.



4. ábra: huBERT + LSTM + sűrű multiklasszifikációs hálózat konfúziós mátrixa

A konfúziós mátrix (lásd 4. ábra) mind a 10 lehetséges kimenetre megmutatja hogyan viszonyulnak a modell által készített előrejelzések az email tényleges kategóriájához képest. Minden lefutás során tisztán látszott, hogy az „Allergia” kategória besorolása okozott problémát a modellnek. Az allergia osztály nehézségét az adja, hogy gyakran csak egy bizonyos összetevőre való érzékenység szerepel a törzsszövegben, az

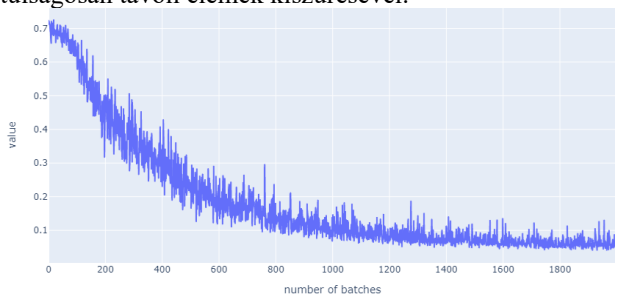
allergia szó maga nem fordul elő és ezzel összefüggésben új étrend kérése vagy a speciális étlap keresése, esetleg lemondás is fellelhető ezekben a levelekben.

4.3 OSL sziámi hálózat két bemenettel és huBERT + sűrű (Dense) osztályozóval

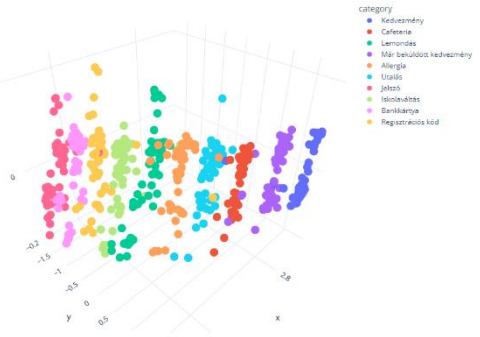
A betanítási loss értéke a kezdeti 0,8 körüli értékről 0,4-re csökkent, a legjobb lefutás pontossága 94%-on állt meg a betanító halmazon, miközben a validációs pontosság 82% volt. A huBERT + sűrű hálózat kombinációjából adódó túlillesztés megmaradt és jelentősen nőtt a véletlenszerű párok miatt a loss görbék volatilitása. A sűrű háló korlátozott képességei ellenére az osztályok rendezése elkezdődött, azonban a legtöbb kategóriának sok elszórt, a saját síkjától távoli pontja van.

4.4 OSL sziámi hálózat 2 bementtel és huBERT + LSTM + Dense osztályozóval

A betanítási loss értéke a kezdeti 0,7 körüli értékről 0,063-ra csökkent, a legjobb lefutás pontossága 96,88%-on állt meg a betanító halmazon, miközben a validációs pontosság 94% volt (5. ábra). A loss függvények volatilitása jelentősen csökkent és a betanítás során végig közel mozognak egymáshoz, azaz nem történt túlillesztés. Az OSL áttranszformálja az eredeti multiklasszifikációs problémát egy bináris páronkénti összehasonlítási feladattá. A hasonlósági értékekre támaszkodva generáltunk egy minden osztályt pontosan egyszer magába foglaló támogató halmazt véletlenszerűen kiválasztott 1-1 db e-maillel. A legnagyobb hasonlósági érték mutatja meg, hogy melyik kategóriába tartozik az adott e-mail. A teljes bemeneten 200 iterációval elvégzett támogató halmaz alapú kiértékelés pontossága 94,5% lett. A 'plotly' modul segítségével a vektorokat 3D interaktív grafikonon ábrázoltuk (6.ábra): mindegyik kategória jól látható módon elkülönül, síkokba rendeződik és viszonylag kevés olyan pont van, mely a saját osztályától távolabb helyezkedik el. A vizuális megjelenítés megerősíti a betanítás végén mért eredményeket. Néhány különálló, eltérő színű pontot még mindig látunk, amik akár egy másik kategória pontjai közé kerültek. Ezeket külön megvizsgáltuk és több esetben is kiderült, hogy emberi félrecímkézésről van szó vagy nem odaillő témakörű levélről. Ezt a fajta adattisztítást a jövőben automatizálni is lehet az adott kategória sűrűsödésétől túlságosan távoli elemek kiszűrésével.



5. ábra: OSL huBERT + LSTM + Dense betanítási loss lefutása



6. ábra: OSL huBERT + LSTM + sűrű háló belső reprezentáció 3D ábrázolása

4.5 Hálózatok eredményeinek összehasonlítása

Az eredmények jobb kiértékelése érdekében klasszikus, kevesebb erőforrást igénylő gépi tanuló eljárásokkal (SVM, Naive Bayes stb.) is elemeztük az adathalmazt. Tokenizálás, szótövezés, stop szavak kiszűrése után alkalmaztuk a TF-IDF módszert és kapott eredményhalmazon 10-szeres keresztvalidációval futtattuk az algoritmusokat. Az eredmények a korábban bemutatott 1. táblázatban láthatók.

Az eredmények igazolják a feladat megoldására egymás után elkészült modellek folyamatos javulását a legfontosabb metrikák, a validációs pontosság, a validációs loss és az F1 pontszám mentén (2. táblázat).

2. Táblázat: Az egyes hálózatok jellemzőinek és eredményeinek összehasonlítása

Modell	Iterációk	Tanítható paraméter	Tanítási pontosság	Validációs pontosság	F1 pontszám	Tanítási loss	Validációs loss
huBERT + Dense multiklasszif.	20 epoch	3 146 442	99,78%	68,6%	60,82%	0,007	1,41
huBERT + LSTM + Dense multiklasszif.	20 epoch	213 898	96,4%	80%	80,24%	0,36	0,76
OSL huBERT + Dense	2000 batch	786 465	Hasonlóság: 94%	Hasonlóság: 82% Támogató. halmaz: 88,5%	88,23%	0,42	0,61
OSL huBERT + LSTM + Dense	2000 batch	215 361	Hasonlóság: 96,88%	Hasonlóság: 94% Támogató. halmaz: 94,5%	98%	0,063	0,23

Az első modell túlillesztett betanítása jól látható: több, mint 30%-kal alacsonyabb a validációs pontosság a betanító halmazon mért értéknél. Az LSTM réteg beillesztése sikeresen orvosolta a hibát, a betanítási pontosság csak minimálisan esett vissza, ellenben 12%-ot nyert a rendszer a validációs metrikán úgy, hogy a validációs loss a felére csökkent. Az OSL modellek támogató halmazon történő kiértékelése a korábbiakkal összehasonlítható mérés, hiszen az eredeti feladatra adja vissza a teljesítményt. A lefutás során kapott értékek azonban az összehasonlítási feladatra vonatkoznak, amit a 2. táblázatban tüntettünk fel. Végül a legjobb eredmény 94,5% és 0,23 volt, ahol utóbbi érték az első loss érték kevesebb, mint ötöde. Az F1 pontszám a multiklasszifikációs modellek esetén kisebb vagy egyenlő a pontossággal, míg az OSL modellek esetén felülmúlja a hasonlósági pontosságot.

Az OSL architektúrára áttérés egy nagyobb volatilitást is behozott a rendszerbe, hiszen a véletlen hatás felerősödött a párba rendezés és a batchek összeállítása során. A fejlesztés előrehaladásával a tanulási rátát úgy állítottuk be, hogy minél kevesebb iteráció alatt eljusson a loss függvény az ellaposodott szakaszába. Az utolsó, LSTM réteget felhasználó modell látványosan jobban kordában tartotta az ingadozásokat ezen a szakaszon. Úgy véljük, az LSTM réteg hatékonysága és a tanulási ráta beállítása elégséges betanítási idővel kulcsfontosságú a hálózat által elért eredményben.

Az első két multiklasszifikációs modell bár osztályonként térségekre bontotta a teret, az ezek közti átfedés hibás kategorizáláshoz vezetett. Az „Allergia” kategória elkülönítése bizonyult a legnehezebb feladatnak. Az OSL architektúrák síkokat alakítottak ki és a síktól távol eső pontok száma, ezáltal a keveredés jelentősen csökkent. Érdekesség, hogy kiderült, a belső reprezentáció kinyerése és ábrázolása felhasználható az emberi pontatlanságok azonosítására is.

5 Összefoglalás és további kutatások

A cikkben bemutatott gyakorlati kutatás célja egy valós ügyfélszolgálati probléma, egy adott céghez beérkező nagyszámú emailek minél pontosabb kategorizálásának a megoldása volt. A kiindulási pont a cég által megadott 10 kategória és 508 előre besorolt email volt. A kutatás során számos különböző megoldás szisztematikus vizsgálatán volt a hangsúly. Nem csak különböző MI alapmodellek, hanem eltérő architekturális megoldások vizsgálatára is sor került – a modellek paramétereinek változtatása mellett. Mint az eredmények igazolták, egy ilyen lépcsőzetes építkezéssel a közbülső eredmények figyelembevételével jelentősen javíthatók a megoldás kimeneti paraméterei. Nemcsak a huBERT sokoldalú felhasználhatósága derült ki, de a Sziámi architektúra ’one-shot-learning’ ígérete is reálisnak bizonyult.

A fent bemutatott legjobb megoldás valós helyzetben való tesztelése (és ezzel a bevezetés előkészítése) jelenleg folyik, illetve további architektúrák kiépítése és az osztályozás több-címkés (multi-label) módban történő vizsgálata szerepel a tervekben.

Köszönetnyilvánítás

A cikk szerzői köszönetüket fejezik ki a Netkir Zrt., az E-Menza licencet birtokló cég irányába, valamint a Webra Kft. felé, amely az E-Menza rendszer fejlesztését végzi.

Bibliográfia

Borg, A., Boldt, M., Rosander, O., Ahlstrand, J. : E-mail classification with machine learning and word embeddings for improved customer support. Neural Computing and Applications, 33(6), 1881–1902 (2021).

Cassandra, C., Hartono, S., Karsen, M. Online Helpdesk Support System for Handling Complaints and Service. 2019 International Conference on Information Management and Technology (ICIMTech), pp. 314-319 (2019).

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. . BERT: Pre-training of deep bidirectional transformers for language understanding. ArXiv Preprint ArXiv:1810.04805 (2018).

Ferrario, A., Loi, M.: Algorithm, machine learning and artificial intelligence. Elgar Encyclopedia of Technology and Politics, 37(1), 135 (2022).

Finardi, P., Viegas, J.D., Ferreira, G.T., Mansano, A.F., Caridá, V.F.: BERTaก: Itaก BBERT for digital customer service. ArXiv Preprint ArXiv:2101.12015 (2021).

Hardalov, M., Koychev, I., Nakov, P.: Towards automated customer support. International Conference on Artificial Intelligence: Methodology, Systems, and Applications, pp. 48–59 (2018).

Hardalov, M., Koychev, I., Nakov, P.: Machine reading comprehension for answer re-ranking in customer support chatbots. Information, 10(3), 82 (2019).

Indurkhya, N., Damerau, F.J.: Handbook of natural language processing. Chapman and Hall/CRC (2010).

Lu, Z., Du, P., Nie, J.-Y.: VGCN-BERT: augmenting BERT with graph embedding for text classification. European Conference on Information Retrieval, pp. 369–382 (2020).

Mnasri, M.: Recent advances in conversational NLP: Towards the standardization of Chatbot building. ArXiv Preprint ArXiv:1903.09025 (2019).

Nemeskey, D.M.: Introducing huBERT. In Berend G., Gosztolya G. Vincze V. (Eds.), XVII. Magyar Számítógépes Nyelvészeti Konferencia, pp. 29-36. Szegedi Tudományegyetem (2021)

Netkir Zrt.: Menza, E-Menza. www.e-menza.hu (2021).

Nguyen, T.: Potential effects of chatbot technology on customer support: A case study. Master's thesis. Http://urn.fi/URN:NBN:fi:aalto-201906233987, Aalto University (2019).

Saberi, M., Hussain, O.K., Chang, E.: Past, present and future of contact centers: A literature review. Business Process Management Journal, 23(3), 574-597 (2017).

Yang, Y., Cui, X.: Bert-enhanced text graph neural network for classification. Entropy, 23(11), 1536 (2021).

Zendesk: The Zendesk Benchmark-Your Prescription for Optimizing Customer Service. https://d16cvnquvjw7pr.cloudfront.net/resources/whitepapers/Zendesk_WP_benchmark.pdf (2020).