Charmen ELECTRA - Tokenizációmentes diszkriminatív nyelvi modellezés

Ficsor Tamás[1], Cserháti Réka[1], Novák Attila[2], Mihajlik Péter[3], Zainkó Csaba[3], Berend Gábor[1]

[1]Szegedi Tudományegyetem, Informatikai Intézet
{ficsort,cserhatir,berendg}@inf.u-szeged.hu
[2]Pázmány Péter Katolikus Egyetem, Információs Technológiai és Bionikai Kar
novak.attila@itk.ppke.hu
[3]Budapesti Műszaki és Gazdaságtudományi Egyetem,
Távközlési és Médiainformatikai Tanszék
{mihajlik,zainko}@tmit.bme.hu

Kivonat Napjainkban a természetesnyelv-feldolgozás területén használt neurális modellek többsége előre definiált szótöredékekből kialakított szótárakkal dolgozik. A kötött szótár használatának eredményeképp az ezekre építő modellek érzékenyek a zajra, doménadaptációjuk költségesebb lehet, ezen felül többnyelvű modellek építése esetén a szótár mérete drasztikusan megnőhet. Ezen problémák orvoslására egy tokenizálómentes ELECTRA architektúrát mutatunk be, amely a Charformer blokkot alkalmazza a tokenizáló modul kiváltására. A modell $\sim$ 17%-kal kevesebb paramétert tartalmaz, mint a fix szótárral rendelkező társa. Továbbá azonos körülmények között tanított társánál szignifikánsabban jobb eredményt ér el az OpinHuBank adathalmazon. Kutatásunk további eredménye, hogy a huBERT modell finomhangolásával a szentimentosztályozás területén az OpinHuBank adatbázison a korábbi legjobb eredményt meghaladó teljesítményt értünk el.
Kulcsszavak: Transzformer, Tokenizáció, Vélemény és Érzelem Analízis

1. Bevezetés

A kontextusfüggő vektorokat előállító architektúrák (Peters és mtsai, 2018; Devlin és mtsai, 2019) megjelenése óta lényegében egyetlen komponenssel, end-to-end módon végezhető el a természetes nyelvi objektumok modellezése, dedikált elemző, illetve előfeldolgozó komponensek (pl. szófaji elemző, dependenciaelemző, stb.) alkalmazása nélkül. Az input szekvenciák teljesen end-to-end feldolgozásának a tokenizálás szab gátat. A tokenek előállításához szükséges szótárt előfeldolgozási lépésként határozták meg, általában valamilyen jól bevált statisztikai módszerrel (Kudo, 2018; Schuster és Nakajima, 2012; Sennrich és mtsai, 2016). Így a modell tanítása során lerögzített szótöredékek alakultak ki. Ezáltal viszont domain-specifikus feladatok (Gong és mtsai, 2018) és alacsony erőforrással rendelkező nyelvek (Hu és mtsai, 2020) modellezése nehézkessé vált. A

tokenizálók érzékennyé teszik a modelleket az inputban található esetleges elírásokra, illetve többnyelvű modellek esetén erőforrásigényes lehet ezen szótárak tárolása és tanítása.

A bemenet karakterszintű modellezésére voltak már korábbi törekvések, de azok csupán a meglévő tokenizálások karakterekre bontására szorítkoztak (Ma és mtsai, 2020). A közvetlenül karakterek lánculatán történő modellezés költséges feladat is lenne, mivel a bemeneti szekvenciahossz drasztikusan meg tudna növekedni ezáltal. Ennek a problémának a megoldására a Charformer (Tay és mtsai, 2021) ad egy lehetséges alternatívát, ahol kontextusfüggő byte szintű reprezentációt alkalmazunk alulmintavételezéssel. Egy másik megközelítést a CANINE (Clark és mtsai, 2021) nyújt, ahol a bemeneti karaktereket hash-beágyazással reprezentálják.

Ebbe a kutatási trendbe illeszkedve cikkünkben bemutatunk egy új, az ELECTRA (Clark és mtsai, 2020) architektúrára épülő, tokenizálómentes modellt, ahol tokenizálás helyett a Charformer modult alkalmazzuk. Ezek együttes működéséhez még további strukturális változásokat eszközölünk a transzformer működésében. Az előtanítást a Hungarian Webcorpus 2.0 Wiki alkorpuszán végezzük, míg a kiértékelést az OpinHuBank adathalmazon. A tanításhoz szükséges kódot is közzétesszük a Github repozitóriumunkban[1] az előtanított súlyokkal együtt.

2. Kapcsolódó irodalom

A kontextualizált nyelvi modellek elterjedése az elmúlt években nagy mértékben előrelendítette a természetesnyelv-feldolgozási feladatok megoldását. Az egyik első ilyen nyelvi modell a transzformer architektúrára (Vaswani és mtsai, 2017) épülő BERT (Devlin és mtsai, 2019) volt. A BERT kiemelkedő teljesítményt ért el számos diverz nyelvfeldolgozási feladaton. A transzformerre épülő nyelvi modellek családja az évek során egyre bővül. Különböző megközelítéseket láthattunk: voltak, akik a meglévő BERT teljesítményén igyekeztek javítani a modell tanítási körülményeinek megváltoztatásával (Zhuang és mtsai, 2021), míg mások a tanítás erőforrásigényét igyekeztek csökkenteni (például „desztilláló modelltömörítés" segítségével; Sanh és mtsai (2019)). Egy, a BERT-től eltérő működést képviselő modell az ELECTRA (Clark és mtsai, 2020). Clark és mtsai (2020) a BERT autoenkóder-stílusú generatív tanítása helyett azt javasolták, hogy a versengő modelleket alkalmazó paradigma szellemében egy kis generatív háló mellett egy diszkriminátort is tanítsunk, a diszkriminátor feladatául a tanítás során azt a bináris osztályozási feladatot szabva, hogy az minél pontosabban legyen képes beazonosítani a generátor által megváltoztatott részeit az inputnak. Ez a megközelítés gyorsabb és jobb konvergenciát mutatott a BERT által nyújtottaknál, tehát a tanítás erőforrásigényét sikerült csökkenteni, még ha a standardnak számító benchmark feladatokon nyújtott eredményessége némileg el is maradt elődjétől.

Nem kellett sokat várni a BERT alapú nyelvi modellek magyar nyelvű variánsainak megjelenésére sem. Nemeskey (2020, 2021) bemutatta a huBERT-et,

[1] `https://github.com/ficstamas/charmen-electra`

amely a Hungarian Webcorpus 2.0-n előtanított BERT Base konfigurációjú modell. Ezzel párhuzamosan jelent meg a HILBERT (Feldmann és mtsai, 2021) is, amely a huBERT-től eltérően a BERT Large konfigurációt használja. Ennek előtanításához egy eltérő adathalmaz-együttest használtak. Bár eddig is léteztek a magyar nyelvet támogató többnyelvű modellek (mBERT, Devlin és mtsai (2019); XLM, Conneau és Lample (2019); XLM-RoBERTa, Conneau és mtsai (2020)), egynyelvű változataik jobban teljesítenek az Ács és mtsai (2021) által vizsgált feladatokon.

Ezen modellek nagy hátránya, hogy fix szótártól függenek. A szótárak előállításához számos módszer létezik – Unigram (Kudo, 2018), WordPiece (Schuster és Nakajima, 2012), Byte-Pair Encoding (Sennrich és mtsai, 2016) – azonban előbbutóbb az alkalmazás során találkozni fogunk szótáron kívül eső szótöredékekkel (főleg doménspecifikus feladatoknál), ezenfelül ez megnöveli a nyelvi modell paraméterhalmazát is. További problémája még, hogy a bemeneti sztring kismértékű változtatása sokszor erősen hat a tokenszekvenciára, tehát a modell érzékeny az elírásokra. Ezen problémák többnyelvű modellek esetén sokkal nyilvánvalóbbak, de kialakulhatnak egynyelvű reprezentációk esetén is.

Számos egyéb megközelítés mellett Clark és mtsai (2021) ennek megoldására tettek javaslatot a CANINE architektúrával. Ezen modell bemeneti azonosítói a karakterek UTF-8 kódolása szerinti reprezentációk. Ebből a kontextuális karakterreprezentációkat további hash-beágyazások (Svenstrup és mtsai, 2017) alapján állítja elő és továbbítja a transzformernek. Egy másik megoldás a Tay és mtsai (2021) által bemutatott Charformer blokk. Az egyik legnagyobb különbség a két módszer között, hogy a Charformer byte-szinten kódolja a bemenetet, míg a CANINE karakter (hash) szinten. Továbbá a Charformer a karakterek fix kiterjedésű szomszédságát is figyelembe veszi.

3. Architektúra

A fejezetben bemutatandó új architektúra két koncepcióra épít. Az első, hogy a modell legyen tokenizációfüggetlen, amelyet a Charformer blokk (Tay és mtsai, 2021) alkalmazása biztosít. Ezenfelül a hatékony konvergencia érdekében a tanítás menete ne generatív, hanem diszkriminatív módszer mentén történjen, erre pedig az ELECTRA modell biztosítja az alapot. Az ezen alapokra ültetett modellünket Charmen ELECTRA-nak (**kar**akteralapú, tokenizáció**men**tes **ELECTRA**) kereszteljük, amelyet Charmen-E-nek rövidítünk az ábrákon.

3.1. Charformer

Ebben a részben a Tay és mtsai (2021) által definiált Charformer architektúra működését mutatjuk be. Első lépésként lerögzítjük az $\mathcal{E} \in \mathbb{R}^{V \times d}$ szótármátrixot, ahol V a tokenek száma, d pedig a dimenziók száma. Esetünkben $V = 263$ mivel 256 byte-ot kódolunk, és fenntartunk további 7 vektort speciális tokenek eltárolására (*[PAD]*, *[MASK]*, *[CLS]*, további négy tokent végül mi nem hasznosítottuk). Így a bemenő szöveget a karakterek UTF-8-as reprezentációjával, mint byte-szekvenciával kódoljuk, amit $X \in \mathbb{R}^{n \times d}$-vel jelölünk.

Ezt követően a beágyazásunkra egydimenziós konvolúciót alkalmazunk a karakterek mentén, hogy globális információt szerezzünk az egyes pozíciókról. Az így kapott karakterbeágyazást tovább alakítjuk egy olyan látens reprezentációba, ahol a szótöredékeket reprezentáljuk legfeljebb M méretű kiterjedésben. Ehhez egy csúszóablakos átlagolófüggvényt vezetünk be $F : \mathbb{R}^{b \times d} \to \mathbb{R}^d$, ahol $b \in \{1, 2, \ldots, M\}$ a blokk mérete. Így

$$X_b = \left(F(X_{i:i+b}); F(X_{(i+s):(i+s)+b}); \ldots\right),$$

definiálja az i karakterpozícióhoz és b méretű szótöredékblokkhoz tartozó reprezentációt, $s = b$ lépésköz mentén. Ez alapján minden karakterpozícióhoz (i) meg tudjuk mondani, hogy melyik szótöredék (b méretű) definiálja azt. Ehhez vegyük az $F_R : \mathbb{R}^d \to \mathbb{R}$ lineáris transzformációt. Ebből adódik, hogy $X_{b,i}$ b blokkhoz tartozási értéke:

$$p_{b,i} = F_R(X_{b,i}).$$

Továbbá bevezetünk egy konszenzus modult is, hogy az egyes pozíciók közötti döntésről legyen információnk. Úgy tekinthetünk rá, mint egy blokkok közti self-attentionre, ami a következőképpen van formalizálva:

$$\hat{P} = softmax(PP^T)P.$$

A szótöredékek keverékét alkotó látens reprezentációt pedig a következőképpen tudjuk meghatározni:

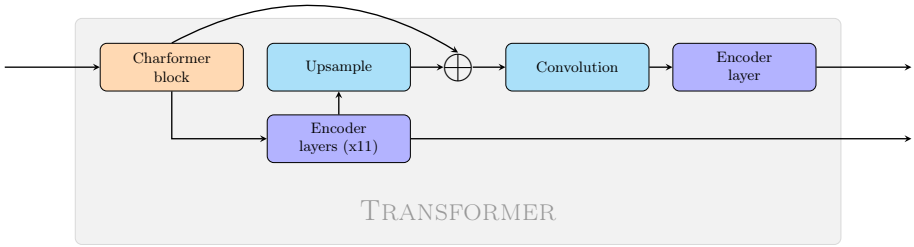
$$\hat{X}_i = \sum_{b=1}^{M} \hat{P}_{b,i} X_{b,i}. \tag{1}$$

Az így előállt reprezentáció redundáns információt tartalmazhat, mivel az információ alacsony lexikális szinten van reprezentálva. Ehhez egy $F_D : \mathbb{R}^{n \times d} \to \mathbb{R}^{\frac{n}{d_s} \times d}$ átlagoláson alapuló összegzőfüggvényt alkalmazunk, amely d_s faktorral fogja a bemenet hosszát csökkenteni. Az így előállt reprezentációt $\tilde{X}$ fogja jelölni.

3.2. ELECTRA

Az eredeti ELECTRA modell egy generátor- és egy diszkriminátor hálóból áll. Mindkét háló transzformer architektúrájú. Lényeges különbség a komplexitásukban (paraméterszám) és a feladataikban van. A generátort jóval kevesebb paraméter alkotja, mint a diszkriminátort. Ennek egyik oka, hogy a generátort az előtanítás után eldobjuk, a másik, hogy komplexitásának növelése nagyon megnövelné a tanítás költségét. Ezenfelül az erre irányuló kísérletek során az az eredmény adódott (Clark és mtsai, 2020), hogy a generátor komplexitásának növelése nem vezet a diszkriminátor teljesítményének javulásához.

A generátor feladata megegyezik a maszkolt nyelvmodellezéssel (MLM). Bemenetként a generátormodell maszkolt szöveget kap, és a maszkolt fragmensek helyére jól illeszkedő szövegrészleteket javasol. A diszkriminátor bemenete egy olyan megváltoztatott szekvencia lesz, amelyet a generátor kimenetéből állítunk



1. ábra: A transzformer működésének javasolt módosítása, ahol az alsó kimenet egy alulmintavételezett reprezentációt ad, a felső pedig a bemeneti szekvenciával azonos méretűt. A $\oplus$ a 3.3 szekcióban bemutatott konkatenációs művelet.

össze. Az eredetileg maszkolt elemek helyére a generátor által javasolt lehetséges jelöltekből a Gumbel-Softmax eloszlás szerint mintavételezünk. Az így előállított pozíciókat korrupt pozícióknak nevezzük, amennyiben a korrumpált token nem egyezik meg az eredeti bemeneti tokennel. A diszkriminátor feladata eldönteni a szekvencia minden pozíciójáról, hogy az ott szereplő token korrumpálva lett-e vagy sem.

Feltűnhet a probléma, hogy ha a tokenizálást Charformer blokkal helyettesítjük, akkor (a bemenet és kimenet hosszának változása miatt) nem vagyunk képesek se a maszkolt nyelvi modellezés (MLM) hibáját meghatározni, se mintavételezni. Ennek megoldására a generátort byte-szintű maszkolt nyelvmodellezésre és a maszkolt byte-ok visszaállítására tanítjuk be, míg a diszkriminátor az egyes byte-ok korrumpáltságát fogja ellenőrizni. Ehhez helyre kell állítani a transzformer kimenetét az eredeti szekvenciahosszra, amihez a következő módosításokat eszközöljük.

3.3. Transzformer

Emlékeztetőképpen a $\hat{X}$ jelölte a beágyazásunkat (1) a Charformer alkalmazásával, $\tilde{X}$ pedig ennek a d_s faktorral arányosan alulmintavételezett változatát. A most bemutatandó folyamatot az 1. ábrán is szemléltetjük.

Elsőként az alulmintavételezett reprezentációnkat továbbadjuk $L-1$ encoder rétegnek. Ennek a kimenete használható egyéb feladatokhoz – például szekvenciaosztályozásnál – azonban az előtanításhoz nekünk nem megfelelő.

$$h^{(1)}_{\tilde{X}} = Encoder_1(\tilde{X}), \quad h^{(i)}_{\tilde{X}} = Encoder_i(h^{(i-1)}_{\tilde{X}}),\ i \in \{2, \ldots, L-1\}$$

Ahhoz, hogy a szekvencia hosszát visszaállítsuk, a Clark és mtsai (2021) által javasolt módszert alkalmazzuk. Először a kiemenetet felülmintavételezzük olyan módon, hogy minden vektort d_s-sel arányos mennyiségben megismétlünk egymás után. Ezt követően az eredeti beágyazást ($\hat{X}$) és a felülmintavételezett reprezentációt összekonkatenáljuk a $\oplus : \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times 2d}$ művelettel. A

felülmintavételezett rejtett reprezentáció a következőképpen adódik:

$$h_{up} = Conv(\hat{X} \oplus \text{UPSAMPLE}(h_{\hat{X}}^{(L-1)}, d_s), w),$$

ahol w a konvolúciós kernel szélessége, ami a mi esetünkben mindig azonos a maximális blokkmérettel (M). Továbbá a konvolúcióra a következő teljesül: $Conv : \mathbb{R}^{n \times 2d} \to \mathbb{R}^{n \times d}$.

Erre a reprezentációra még egy utolsó Encoder réteget alkalmazunk, hogy a skálázott reprezentációt finomítsuk.

$$h_{up}^{(L)} = Encoder_L(h_{up})$$

4. Konfiguráció

4.1. ELECTRA

Ahhoz, hogy a modellünk teljesítménye összehasonlítható legyen a jelenlegi tokenizálóeljárásokat alkalmazó modellekével, a huBERT[2] tokenizálójával is előtanítunk egy ELECTRA modellt. A generátor[3] és a diszkriminátor[4] a Huggingfaceen közzétett ELECTRA Base konfiguráció szerint lett inicializálva. Továbbá a maximális bemeneti szekvencia hossz 256-re van állítva. Erre a modellre egyszerűen csak *ELECTRA*-ként fogunk hivatkozni.

4.2. Charmen ELECTRA

Az 3. szekciókban bemutatott modellt az ábrákon csak *Charmen-E*-ként fogjuk említeni. Ezen kívül, ha a felülmintavételezett kimenetet használjuk, akkor azt külön jelezni fogjuk. A transzformer eredeti moduljait (encoder rétegeket, beágyazásokat) a Huggingface-en is található konfiguráció szerint inicializáltuk (beleértve mind a generátor-, mind a diszkriminátor blokkot). A tokenizáló egységen belül két paraméter változtatásának hatását vizsgáljuk: a maximális blokkméretét ($M \in \{4, 6\}$), és az alulmintavételezési arányét ($d_s \in \{2, 4\}$). d_s-sel arányosan a maximális szekvenciahossz is változik, méghozzá úgy, hogy az alulmintavételezett hossz megegyezzen az ELECTRA esetén alkalmazott értékkel (256).

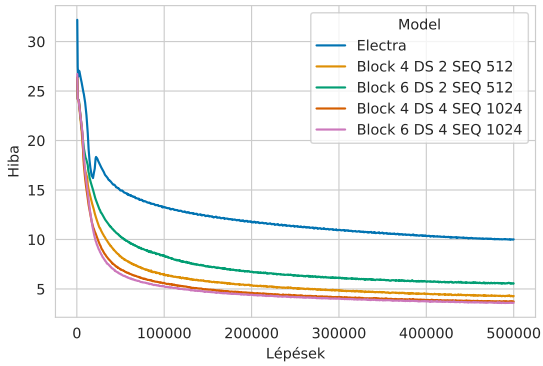
5. Előtanítás

Az előtanítás során az ELECTRA modell két hibatag kombinációját veszi figyelembe. A generátor paramétereit a maszkolt nyelvi modellezési hiba visszaterjesztésével tanítjuk, ami a keresztentrópiával van kifejezve. A diszkriminátor tanításához pedig a korrumpált szövegrészek felderítésére vonatkozó hibát használjuk, ami bináris keresztentrópiával van meghatározva. Feltűnhet, hogy a generátor hibatagja könnyen elnyomhatja a diszkriminátor hibáját (hiszen míg a

[2] `https://huggingface.co/SZTAKI-HLT/hubert-base-cc`
[3] `https://huggingface.co/google/electra-base-generator`
[4] `https://huggingface.co/google/electra-base-discriminator`

	Batch-méret	Lépések	Tanítási idő	M	d_s	Szekvenciahossz	Hiba	$\|\theta\|$
ELECTRA	16	500 000	≈150 óra	–	–	256	10.02	120M
Charmen ELECTRA	16	500 000	≈78 óra	4	4	1024	3.73	101M
Charmen ELECTRA	16	500 000	≈78 óra	6	4	1024	3.56	101M
Charmen ELECTRA	16	500 000	≈63 óra	4	2	512	4.27	98M
Charmen ELECTRA	16	500 000	≈63 óra	6	2	512	3.59	98M

1. táblázat. Az előtanított modellek konfigurációja, és az előtanítással járó idő, illetve hiba. d_s az alulmintavételezési arányt, M pedig a maximális blokkméretet jelöli.



2. ábra: Az egyes modellek előtanítása során a hiba változása. ELECTRA a huBERT tokenizációs modellt takarja. A további 4 konfiguráció a Charmen ELECTRA paramétereit írja le. Block a maximális blokkméret (M), DS az alulmintavételezési arány (d_s), és `SEQ` a maximális bemeneti szekvencia hossz.

generátor egy sokosztályos, addig a diszkriminátor csupán egy bináris osztályozási feladatot lát el), ezért bevezetünk egy súlytényezőt az egyes hibatagokra, Clark és mtsai (2020) nyomán:

$$\mathcal{L}(x, \theta) = \lambda_1 \mathcal{L}_{Gen}(x, \theta_G) + \lambda_2 \mathcal{L}_{Disc}(x, \theta_D),$$

ahol (λ_1, λ_2) a két hibataghoz tartozó súly. Az eredeti cikkben javasolttal megegyező módon az $(1, 50)$ értékeket használtuk a λ súlyparaméterekre.

A modelleket a Hungarian Webcorpus 2.0[5] (Nemeskey, 2020) Wikipedia-alkorpuszán tanítottuk, 500 000 lépésen keresztül összességében 16-os batch-mérettel (8-as batch-méret/GPU). A tanítás során $8 \cdot 10^{-5}$ tanulási rátát használtunk AdamW optimalizálóval ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda = 0.01$), és lineáris tanulásiráta-ütemezőt 15 000 lépéses warmuppal. Az így előállított modellek konfigurációi részletesebben az 1. táblázatban láthatóak.

Az előtanítás során a hiba változását a 2. ábrán láthatjuk, ez magába foglalja együttesen a generátor és a diszkriminátor hibáját is. Az ábrán az ELEC-

[5] `https://hlt.bme.hu/en/resources/webcorpus2`

TRA és Charmen ELECTRA modellek hibája van feltüntetve, ahol a Charmen ELECTRA modellváltozatok a konfigurációs értékeik szerint van megjelenítve. Az ábrán belül a `Block` a maximális blokkméretet (M), `DS` az alulmintavételezési arányt (d_s), a `SEQ` pedig a maximális bemeneti szekvenciahossz értékét jelöli. A Charmen ELECTRA hibaértéke első ránézésre jóval kedvezőbbnek tűnhet, azonban mivel az MLM feladat során az ELECTRA nagyjából 32 000 lehetséges kimenet közül választ, míg a Charmen ELECTRA csupán 263-ból, így az utóbbi feladata könnyebbnek is tekinthető. A Charmen ELECTRA modellek körében a nagyobb bemeneti szekvenciahossz kedvezőbbnek tekinthető (a modellezőképesség szempontjából), ugyanakkor az ábráról leolvasható, hogy a hibacsökkenés szempontjából a két vizsgált szekvenciahossz, és a két vizsgált blokkméret esetében ellentétesen viselkednek a modellek.

6. Finomhangolás

A Charmen ELECTRA esetén számos kérdést felvet, hogy mit tekinthetünk egy tokennek, hiszen a Charmen ELECTRA által használt egymás utáni karakteregyüttesek akár szóhatárokon is átívelhetnek. Ennél fogva a tokenszintű osztályozási feladatok megvalósítása nem egyértelmű. Figyelembe véve, hogy az ELECTRA és a Charmen ELECTRA más szinten reprezentálja az információt a kimeneten, így összehasonlítási alapnak a teljes szekvenciák osztályozására vonatkozó feladat használata mellett döntöttünk.

A kvalitatív elemzéshez az OpinHuBank[6] (Miháltz, 2013) adathalmazt választottuk. Az adathalmaz érzelmi attitűd (szentiment) osztályozási feladatot definiál. Összesen 10 000 mondat alkotja, melyekhez entitások tartoznak. Ezeket a mondat–entitás-párokat 5 annotátor pozitív, negatív és semleges kategóriába sorolta. A címkét többségi döntés alapján határozzuk meg, és ennek a meghatározása lesz a modell feladata.

A feladatot két felosztásban is megvizsgáljuk követve Hangya és mtsai (2015) munkáját. Az első esetben az összes címkét megtartjuk az osztályozáshoz, a második esetben pedig csak a nem semleges polaritást hordozó címkékkel dolgozunk. Az adathalmazt 70% tanító, 10% validációs és 20% teszthalmazra bontottuk fel.

Tekintve, hogy a modelljeink előtanítása során hiperparaméter-finomhangolást nem végeztünk, így egy stabilabb modell viselkedését is bevontuk az összehasonlításba. Ehhez a Webcorpus 2.0-n és a Wikipedia-alkorpuszon betanított huBERT modelleket is alkalmaztuk a kísérletekbe.

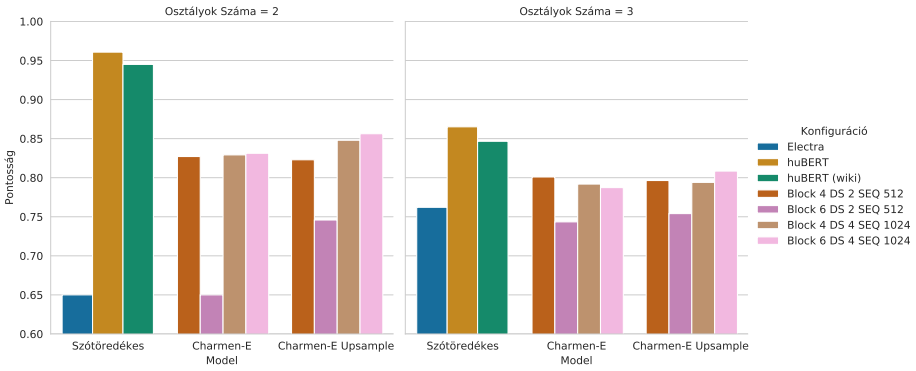
6.1. Hiperparaméterek

A modellek finomhangolását 3 epochon keresztül végeztük. Ez eddigiekhez hasonlóan AdamW optimalizálót használtunk ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda = 0.1$), $5 \cdot 10^{-5}$ induló tanulási rátával. A tanulási ráta értékét lineáris ütemben csökkentettük, és warmupot nem alkalmaztunk.

[6] `https://sites.google.com/site/mmihaltz/resources#h.p_ID_42`

#Lefagyasztott Rétegek	0	2	4	6	8	10	12
HuBERT	0.84	0.83	0.85	0.85	0.84	**0.85**	0.81
ELECTRA	0.74	0.74	0.74	0.73	0.74	0.75	**0.77**
Charmen ELECTRA	0.74	0.78	**0.81**	0.80	0.79	0.76	0.74
Charmen ELECTRA + Upsample	0.79	0.74	**0.81**	0.80	0.79	0.80	0.76

2. táblázat. A teljesítmény változása a validációs halmazon különböző számú lefagyasztott réteg alkalmazása mellett háromosztályos osztályozás esetén.



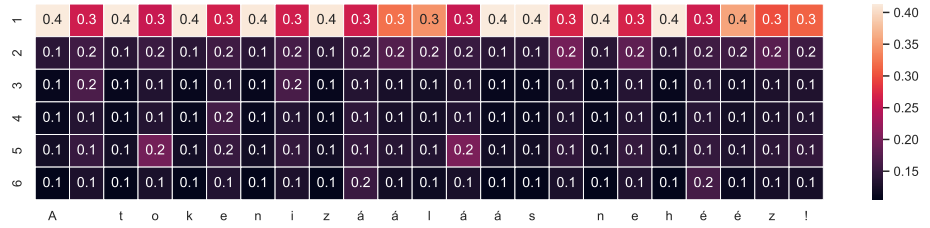
3. ábra: A modellek teljesítménye két- és háromosztályos osztályozási feladatokon. Az utolsó négy konfiguráció a Charmen ELECTRA használatának lehetséges konfigurációira vonatkozik. Block a maximális blokkméret (M), DS az alulmintavételezési arány (d_s), és SEQ a maximális bemeneti szekvencia hossz.

A kísérletek során a rétegek egy részét lefagyasztottuk ezzel csökkentve az túlillesztés esélyét és gyorsítva a tanítást, valamint csökkentve a memóriahasználatot. Az ehhez elvégzett kísérletek eredményeit a 2. táblázatban tüntettük fel. A táblázatban a három osztályt tartalmazó adathalmazon végzett eredmények láthatók a validációs halmazon kiértékelve. Az eredmények alapján a huBERT esetén 10, az ELECTRA esetén 12, a Charmen ELECTRA mindkét lehetséges kimenete esetén 4 réteg lefagyasztásával kaptuk a legjobb eredményeket.

6.2. Eredmények

Az OpinHuBank-en végzett kísérleteink eredménye a 3. ábrán látható. Itt fel van tüntetve a két- (bal) és háromosztályos (jobb) osztályozási eset is. Mindkét ábrán három részre osztva jelenítettük meg az eredményeket. Az első csoportot a szótöredéken alapuló szótárat használó modellek, a másodikat a kimenet visszaskálázása nélküli Charmen ELECTRA-modellek, a harmadikat pedig a kimenet felskálázását használó Charmen ELECTRA modellek alkotják.

A kizárólag a semlegestől eltérő címkéket használó kétosztályos feladat esetében a magyar BERT modellek teljesítettek a legjobban, egyenként $\sim 96\%$ és



4. ábra: Charformer blokk által tanult $\hat{P}$ mátrix, amely az egyes pozíciókhoz a blokkhoz tartozási valószínűségeket rendeli. Egyes karakterek az x-tengelyen duplán jelenhetnek meg. Ennek az oka, hogy két byte-on vannak reprezentálva.

$\sim$ 94%-os teljesítménnyel. Ezeket követi a Charmen ELECTRA-modellek többsége 82% és 86% közötti teljesítménnyel, amelyek mögött kiemelkedően rossz teljesítményt nyújtva kullog az 512-es szekvenciahosszt és 6-os blokkméretet használó modell. A szótöredékeket használó ELECTRA teljesítménye pedig a legrosszabb mind közül. Ugyanezt a tendenciát vehetjük észre a háromosztályos esetben is, annyi különbséggel hogy az ELECTRA modell relatív teljesítménye valamivel jobb lett. Első ránézésre meglepő, hogy a három egyébként legrosszabbul teljesítő modell abszolút teljesítménye javult a nehezebb feladaton, míg a többi modellé érthető módon romlott. A magyarázatot abban láthatjuk, hogy a mondatok $\sim$ 74%-a többségi döntés szerint semleges az OpinHuBank-ben, és a gyengébb modellek a leggyakoribb címkét preferálják.

Bináris és háromosztályos esetben a két huBERT modell teljesített a legjobban, meghaladva az OpinHuBank adatbázison eddig közölt korábbi legjobb eredményeket (Hangya és mtsai, 2015). A tokenizálómentes Charformer-modelleknek a huBERT-énél valamivel rosszabb teljesítménye nem igazán okozott meglepetést. Egyrészt a huBERT egy nagyon jól tanított modell, másrészt az irodalomban korábban vizsgált esetekben (Tay és mtsai, 2021; Clark és mtsai, 2021) is a tokenizálómentes modellek valamivel mindig a fixen töredékesített társaik mögött végeztek. Cserébe viszont – mint ahogy az 1. táblázatban is látható – gyorsabb és kisebb modellt kapunk, ami maga alakítja ki a befoglaló kontextus függvényében a szótöredékeket. A fixen tokenizált ELECTRA-val összevetve, amely azonos körülmények között lett tanítva, mint a Charmen ELECTRA, sikernek könyvelhető el a teljesítménye.

7. Charformer belső reprezentációja

A Charformer blokk által kialakított belső tokenizáció, valamilyen szinten visszafejthető a modellből. Amennyiben visszatekintünk az (1) egyenletre, észrevehetjük, hogy $\hat{P}$ a pozíciók egyes blokkokhoz tartozási valószínűségét reprezentálja. Ennek a $\hat{P}$ mátrixnak a reprezentációját láthatjuk a 4. ábrán. Ezt a mátrixot az 1024 hosszú bemeneti szekvenciahosszal, 6-os blokkmérettel, és 4-es alulminta-

vételezési faktorral rendelkező Charmen ELECTRA modell adta. Látható, hogy a modell a byte-szintű reprezentációt preferálja, ami nem feltétlen meglepő. Az előtanítás során byte-szinten maszkoltunk, és a diszkriminátor is byte-szinten hozott döntést. Valószínűleg ennek köszönhetően alakult ki ez a klaszterezettség.

7.1. Charformer alapú tokenizálás

Amennyiben a $\hat{P}$ mátrixot átmeneti valószínűségeknek tekintjük, úgy képesek vagyunk fix szegmenseket felismerni a szövegben. Ezzel előállítva egy valószínű szegmentációját a bemeneti szövegnek. Erre a Viterbi algoritmust alkalmazzuk, ahol az állapotok valószínűsége az adott korpuszon kiszámított n-gram valószínűségek lesznek. A mi esetünkben $n = \{1, 2, 3, 4, 5, 6\}$ mivel 6 blokkunk van.

A fenti modell segítségével létrehozott reprezentáció alapján az említett algoritmussal előállított tokenizálásra az alábbiakban mutatunk be néhány példát:

- 'A toke', 'nizál', 'ás n', 'ehéz!'
- 'A', ' kutya', ' szav', 'unk er', 'ededé', 'r', 'e töb', 'b', ' felt', 'étel', 'ezés', ' van.'
- 'A', ' biol', 'ógia', ' szót', ' elős', 'zör M', 'icha', 'el Ch', 'r', 'istoph', ' Hanov', ' ném', 'et fil', 'ozófu', 's', ' hasz', 'n', 'álta', ' 1766', '-ban ', 'egyik ', 'k', 'önyve', ' cím', 'ében.'

Vegyük észre, hogy a blokkméretek merev korlátozó tényezőként hatnak az egyes szótöredékek kialakulása során. Tehát a Viterbi-algoritmus nem tud hosszabb szótöredéket összerakni, mint például hogy *'ehéz!'* (az *é* karakter 2 byte-ot tesz ki az `UTF-8` szerinti kódolása miatt).

8. Konklúzió

Cikkünkben bemutattuk egy tokenizálásmentes megközelítését az ELECTRA modellnek. Bár teljesítménye nem éri el a huBERT modellét, tanítása a huBERT tanításához szükséges erőforrásigényének töredéke volt csupán. Így jóval kisebb és gyorsabb modellt kaptunk, amely az azonos körülmények között tanított merev tokenizáláson alapuló ELECTRA modellnél jobban teljesít. Kutatásunk további eredménye, hogy a huBERT modell finomhangolásával az OpinHuBank adatbázison a korábbi legjobb eredményeket meghaladó osztályozási teljesítményt kaptunk.

Köszönetnyilvánítás

A dolgozatban szereplő kutatási eredmények létrejöttét az Innovációs és Technológiai Minisztérium és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatta a Mesterséges Intelligencia Nemzeti Laboratórium keretében.

Hivatkozások

Ács, J., Lévai, D., Nemeskey, D.M., Kornai, A.: Evaluating contextualized language models for hungarian. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia, pp. 15–28. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2021), http://acta.bibl.u-szeged.hu/73354/1/msznykonf_017_015-028.pdf

Clark, J.H., Garrette, D., Turc, I., Wieting, J.: CANINE: Pre-training an efficient tokenization-free encoder for language representation (2021), https://arxiv.org/abs/2103.06874

Clark, K., Luong, M.T., Le, Q.V., Manning, Ch.D.: ELECTRA: pre-training text encoders as discriminators rather than generators. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=r1xMH1BtvB

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 8440–8451. Association for Computational Linguistics, Online (Jul 2020), https://www.aclweb.org/anthology/2020.acl-main.747

Conneau, A., Lample, G.: Cross-lingual language model pretraining. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E.B., Garnett, R. (szerk.) NeurIPS. pp. 7057–7067 (2019), http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining

Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://aclanthology.org/N19-1423

Feldmann, Á., Hajdu, R., Indig, B., Sass, B., Makrai, M., Mittelholcz, I., Halász, D., Yang Zijian, Gy., Váradi, T.: HILBERT, magyar nyelvű BERT-large modell tanítása felhő környezetben. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia, pp. 29–36. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2021), http://real.mtak.hu/120856/1/feldmann21.pdf

Gong, C., He, D., Tan, X., Qin, T., Wang, L., Liu, T.Y.: FRAGE: frequency-agnostic word representation. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (szerk.) NeurIPS. pp. 1341–1352 (2018), http://papers.nips.cc/paper/7408-frage-frequency-agnostic-word-representation

Hangya, V., Farkas, R., Berend, G.: Entitásorientált véleménydetekció webes híranyagokból. In: XI. Magyar Számítógépes Nyelvészeti Konferencia. pp. 343–345. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2015), http://acta.bibl.u-szeged.hu/58936/1/msznykonf_011_227-234.pdf

Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., Johnson, M.: XTREME: A massively multilingual multi-task benchmark for evaluating cross-

lingual generalisation. In: III, H.D., Singh, A. (szerk.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 4411–4421. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/hu20b.html

Kudo, T.: Subword regularization: Improving neural network translation models with multiple subword candidates. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 66–75. Association for Computational Linguistics, Melbourne, Australia (Jul 2018), https://aclanthology.org/P18-1007

Ma, W., Cui, Y., Si, C., Liu, T., Wang, S., Hu, G.: CharBERT: Character-aware pre-trained language model. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 39–50. International Committee on Computational Linguistics, Barcelona, Spain (Online) (Dec 2020), https://aclanthology.org/2020.coling-main.4

Miháltz, M.: OpinHuBank: szabadon hozzáférhetö annotált korpusz magyar nyelvű véleményelemzéshez. In: IX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 343–345. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2013), http://acta.bibl.u-szeged.hu/58859/1/msznykonf_009_343-345.pdf

Nemeskey, D.M.: Natural Language Processing Methods for Language Modeling. Ph.D.-értekezés, Eötvös Loránd University (2020), https://hlt.bme.hu/en/publ/nemeskey_2020

Nemeskey, D.M.: Introducing huBERT. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2021). pp. 3–14. Szeged (2021), http://acta.bibl.u-szeged.hu/73353/1/msznykonf_017_003-014.pdf

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018), https://aclanthology.org/N18-1202

Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR abs/1910.01108 (2019), http://arxiv.org/abs/1910.01108

Schuster, M., Nakajima, K.: Japanese and Korean voice search. In: ICASSP. pp. 5149–5152. IEEE (2012), https://doi.org/10.1109/ICASSP.2012.6289079

Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (Aug 2016), https://aclanthology.org/P16-1162

Svenstrup, D., Hansen, J.M., Winther, O.: Hash embeddings for efficient word representations. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (szerk.) NIPS. pp. 4928–4936 (2017), http://papers.nips.cc/paper/7078-hash-embeddings-for-efficient-word-representations

Tay, Y., Tran, V.Q., Ruder, S., Gupta, J., Chung, H.W., Bahri, D., Qin, Z., Baumgartner, S., Yu, C., Metzler, D.: Charformer: Fast character transformers via gradient-based subword tokenization (2021), https://arxiv.org/abs/2106.12672

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (szerk.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Zhuang, L., Wayne, L., Ya, S., Jun, Z.: A robustly optimized BERT pre-training approach with post-training. In: Proceedings of the 20th Chinese National Conference on Computational Linguistics. pp. 1218–1227. Chinese Information Processing Society of China, Huhhot, China (Aug 2021), https://aclanthology.org/2021.ccl-1.108