OCR-hibák javítása neurális technológiák segítségével

Laki László János[1,2], Kőrös Ádám[1], Ligeti-Nagy Noémi[1,2], Nyéki Bence[1], Vadász Noémi[1], Yang Zijian Győző[1,2], Váradi Tamás[1]

[1]Nyelvtudományi Kutatóközpont
1068 Budapest, Benczúr u. 33.
vezeteknev.keresztnev@nytud.hu
[2]MTA-PPKE Magyar Nyelvtechnológiai Kutatócsoport
1083 Budapest, Práter utca 50/a
vezeteknev.keresztnev@itk.ppke.hu

Kivonat Munkánk során párhuzamos optikai szövegfelismerővel digitalizált (OCR) szövegeken végeztünk utólagos korrektúrát. Kutatásunkban különböző OCR-hibákat tartalmazó szövegeket detektáló és javító modelleket implementáltunk, valamint saját *Silver standard* párhuzamos korpuszt építettünk. Eredményeink azt mutatják, hogy abban az esetben, amikor kizárólag OCR-hibák javítását tűzzük ki célul, modelljeink a Context-based Character Correction (CCC) detekciós modellel való kombinációval a leghatékonyabbak. A létrehozott enkóder-dekóder alapú javító módszereink az OCR-hibák mellett nagy pontossággal javítják a szövegekben található koherenciahibákat (pl.: oldalszámok, elválasztások). Kutatásunk folytatásaként tervezzük a modellek tanítását és tesztelését a *Gold standard* korpuszon is.
Kulcsszavak: OCR-hiba detektálás, OCR-hiba javítás, OCR-hiba korpuszépítés

1. Bevezetés

Napjainkban a mélyneurális-hálózatok sok más tudományterület mellett a nyelvtechnológia területén is felülmúlták az addigi piacvezető rendszerek teljesítményét. Az előtanított nyelvmodellek építésének sarokpontja a nagy mennyiségű és jó minőségű, egy- vagy többnyelvű tanítóanyag megléte. Ilyen korpuszok nem csak az internetes weblapok letöltéséből gyűjthetők, hanem kézenfekvő megoldásként kínálkozik a nyomtatott dokumentumok felhasználása is. Számtalan könyv, folyóirat, újság stb. áll rendelkezésünkre nyomtatott, de nem digitalizált formában. A nyomtatott művek digitalizálása sok emberi erőforrást igényel. Ezen feladatok támogatására kiválóan alkalmas az optikai szövegfelismerők (Optical Character Recognition – OCR) használata. Sajnálatos módon az OCR-technológiáknak vannak korlátai is, és különféle, változatos hibát generálnak a digitalizálás során.

Magyar vonatkozásban az Arcanum Adatbázis Kiadó[1] egyik célkitűzése a kulturális tartalmak nagy tömegű digitalizálása, adatbázisokba rendezése és a szövegek elérhetővé tétele online szolgáltatás révén. Az NYTK és az Arcanum Adatbázis Kiadó közötti együttműködés keretében végzett kutatásunk során különböző módszereket próbáltunk ki a Kiadó OCR-ezett szövegeinek javítására.

2. Kapcsolódó irodalom

Az OCR-ezett szövegek javításának jelentőségét mutatja, hogy az ICDAR konferencia[2] 2017 óta[3] külön versenyt rendez az OCR-hibák javítására. Az évek során a nyelvtechnológia fejlődésével az OCR-javító algoritmusok is szorosan követték az aktuális legjobb eredményeket elért módszereket.

A 2019-es év győztes OCR-javító módszere, a CCC (Context-based Character Correction, Rigaud és mtsai, 2019) egy BERT (Devlin és mtsai, 2019a) modellel kiegészített konvolúciós hálózaton alapuló detekciós, illetve egy kétirányú LSTM (Long short-term memory) rekurrens neurális hálózaton (figyelmi mechanizmussal) alapuló, enkóder-dekóder architektúrájú korrekciós részből áll.

Nguyen és mtsai (2020) a CCC módszert dolgozták át úgy, hogy a detekciós modellhez egy NER (Name Entity Recognition) modellt adaptáltak, míg a korrekciós modellhez nem rekurrens hálózatot használtak, hanem egy neurális gépi fordító rendszert, az OpenNMT keretrendszert (Klein és mtsai, 2017).

Schaefer és Neudecker (2020) szintén egy kétlépcsős módszert alkalmaztak, ahol az első egy detekciós, a második egy korrekciós modell. A detekciós modellük egy kétirányú LSTM hálózaton alapul, ami minden következő karakterre megjósolja, hogy az hibás vagy sem. A korrekciós modell a kutatásukban szintén egy LSTM alapú gépifordítóra épül.

Duong és mtsai (2021) kutatásukban transzformer (Vaswani és mtsai, 2017) alapú gépi fordítással oldották meg a problémát, detekciós modul nélkül. Ehhez viszont olyan modellt kellett tanítaniuk, amelyben egyaránt szerepelnek helyes és OCR-hibás szövegrészek, hogy a modell felügyelet nélkül megtanuljon mindent magától, mind a helyes, mind a hibás részeket. Ehhez automatikus módszerekkel saját, OCR-hibákat tartalmazó tanítókorpuszt hoztak létre. Kétféle módszert alkalmaztak az automatikus OCR-hibák generálására. Első módszerükben összegyűjtötték a lehetséges OCR-hibákat és véletlen módon belegenerálták azokat a helyes szövegbe. Második módszerükkel egy rekurrens neurális hálózaton alapuló gépi fordító modellt tanítottak be egy párhuzamos OCR-hibás korpusz segítségével. A bemenet egy hibátlan szöveg, a kimenet az OCR-hibás párja. Az így létrejött korpuszokkal tanítottak OCR-javító gépifordító modelleket.

Mei és mtsai (2016) foglalkozott azzal az eshetőséggel, ha több OCR-javító modell is rendelkezésre áll, melyek egyenként állítanak jelölteket az adott OCR-hiba javítására. A különböző modellek különböző jelöltjei közti hatékony választásra statisztikai eljárást dolgoztak ki, mely során a jelölteket OCR-tulajdonságaik

[1] https://www.arcanum.com
[2] https://www.icdar.org
[3] https://sites.google.com/view/icdar2017-postcorrectionocr

(Levenshtein távolság, legnagyobb közös szekvencia, gyakoriság, lexikon ellenőrzés stb.) alapján pontozták, majd a kiválasztási feladatot regressziós modellezésként felfogva súlyozták a jelöltek közötti valószínűségi sorrendet.

Kutatásunk során kipróbáltuk a kétlépcsős (detekció + korrekció) és az egylépcsős gépi fordítással történő megközelítést egyaránt, valamint saját párhuzamos korpuszt is építettünk.

3. Korpuszépítés

Az OCR javításához egy párhuzamos tanítókorpuszt építettünk, ahol a forrásnyelvi oldal egy OCR-hibás szöveg, míg a célnyelvi szöveg annak hibátlan párja. Ehhez olyan nagyobb mennyiségű szöveget kellett választanunk, ami elérhető mind elektronikus (helyes), mind OCR-ezett (hibás) változatban. Ezt követően ezeket manuálisan/fél-automatizálva annotáltuk, majd később annotátorokkal javíttattuk. Az annotált korpusz lett a *Silver standard* korpuszunk, míg az annotátorokkal kijavított hibátlan változat a *Gold standard*.

A *Silver és Gold standard* korpusz építéséhez **J**ókai Mór[4], **i**lletve **M**ikszáth[5] (JiM) összes műveit használtuk fel. Választásunk azért esett rájuk, mert ezeknek elérhető a kiadó által elektronikusan közzétett (nem OCR) változata is.

Azonban a probléma az volt, hogy egyetlen fájlban volt az összes Jókai-mű és egyetlen fájlban az összes Mikszáth-mű. Ezért első feladatként a műveket össze kellett párosítani az Arcanum fájlokkal. Szerencsére a fájlokban a művek az Arcanum Kézikönyvtár[6] oldalán található sorrendben voltak, ezért az adott sorrendben a címek alapján írtunk egy szkriptet, amely a címek mentén közel tökéletesen különválasztotta a műveket külön-külön fájlokba. A Mikszáth-művek szétválasztásánál voltak kezelendő esetek, mivel a fájlban a címek és az alcímek külön sorban voltak, míg az adott sorrendlistában ezek egy sorban álltak, ezért ezeket kézzel korrigálni kellett, és az alcímeket a címekkel egy sorba kellett tenni.

3.1. Párhuzamos korpusz építése

Jókai és Mikszáth összes művének külön fájlba való bontása után következett a párhuzamos korpusz építése. A feladat az volt, hogy az OCR-ezett fájlok között megtaláljuk ugyanezeket a műveket, majd fájlszinten hozzájuk rendeljük. A munkát nehezítette, hogy a digitális kiadás csupán a szövegtörzset tartalmazta, míg az OCR-ezett dokumentum a teljes könyvet, beleértve a címoldalakat, tartalomjegyzékeket, valamint a dokumentum végi jegyzeteket. További nehézség volt, hogy egy OCR-ezett kötetben található könyvek vagy novellák nem mindegyike volt megtalálható a digitális kiadásban, valamint ami szerepelt benne, az sokszor nem azonos sorrendben. Ezen túl a JiM művek nem csak egy kiadó fájljaiban voltak, ezért a megtalálásukat egy Sketch engine motorját használó indexált

[4] https://www.arcanum.com/hu/online-kiadvanyok/Jokai-jokai-mor-osszes-muvei-1
[5] https://www.arcanum.com/hu/online-kiadvanyok/Mikszath-mikszath-osszes-muve-2A85B
[6] https://www.arcanum.com/hu/online-kiadvanyok

keresőbe töltöttük, és ennek a segítségével támogattuk. A fájlok párosítását kézi munkával végeztük. Ezzel előállt a művek szintjén a párhuzamos korpusz.

Következő feladatként ezeket a műveket további szegmensekre bontottuk, és kisebb egységű párhuzamosítást végeztünk rajtuk. Ehhez a feladathoz kétféle módszert is alkalmaztunk. Az egyik a mondatszintű szegmentálás (mondat), a másik egy 100 karakteres (100 karakter) egységekre bontott, gördülőablakos szegmentálás.

A művek kisebb egységű párhuzamosítása előtt a szövegeket előfeldolgozás alá vetettük. Első lépésként NFKC normalizálást[7] végeztünk, majd szólista alapján egyesítettük az elválasztott szavakat: ha kötőjel és egy vagy több szóköz választott el két betűfüzért, és két füzér egyesítése a szólistában szereplő szót adott, akkor az elválasztó karaktereket töröltük. Mindenhol töröltük továbbá a feltételes kötőjeleket, a sor elején és szóközök között álló kötőjeleket (U+2010) és a nagykötőjeleket (U+2013) egységesen az Unicode szerinti U+2014 kódú nagykötőjelre cseréltük le. Több egymást követő szóközt egyetlen szóközre cseréltünk, különböző idézőjel-karaktereket (U+201C, U+201D, U+201E) pedig az U+0022 kódú karakterre.

A fenti előfeldolgozó lépésekkel kezeltük mind a nyers OCR, mind a *Silver standard* korpusz alapjául szolgáló szövegeket. A mondatszintű tokenizáláshoz a *quntoken*[8] eszközt használtuk. Ennek alternatívájaként a szövegeket 100 karakter hosszú egységekre szegmentáltuk gördülőablakkal, melyet legalább 25 karakteres lépésekkel csúsztattunk: a lépést 25 karaktertől kezdve addig növeltük, amíg az ablak új szó elejére nem ért. Miután a szövegeket mondattokenizálással, illetve gördülőablakkal szegmentáltuk, a nyers és a standard szöveg egységei közötti megfeleltetést a Python programozási nyelv *difflib*[9] könyvtárának segítségével végeztük el.

	Tanító		Teszt	
	Elektronikus	OCR	Elektronikus	OCR
Mű	Jókai: 77 Mikszáth: 651			
Mondat	653.799		64.318	
Sor (100 karakter)	1.808.794		199.700	
Token	9.683.628	9.782.166	992.642	996.977
Type	554,541	599.327	127.321	130.062
Karakter	57.691.276	57.841.735	5.934.846	5.957.890
Átlagos mondathossz	14,81	14,96	15,43	15,51

1. táblázat. A JiM korpusz tulajdonságai.

Az így létrejött párhuzamos korpuszt felbontottuk 90%-10% arányban tanító- és tesztkorpuszra. Az 1. táblázatban láthatóak a JiM korpusz főbb tulajdonságai.

[7] https://unicode.org/reports/tr15
[8] https://github.com/nytud/quntoken
[9] https://docs.python.org/3/library/difflib.html

3.2. *Silver standard* korpusz előállítása

A párhuzamos korpusz elkészültével következett a *Silver standard* korpusz építése, ügyelve arra, hogy elkülönítsük a teszt- és a tanítóanyagot. A feladathoz vettük a mondatalapú párhuzamos korpuszt és fél-automatikusan annotáltuk a szövegben felmerülő **OCR**, **koherencia** és a kiadások közötti **központozási** különbségekből eredő hibákat.

Fél-automatikus annotáláson a következő eljárást értjük: A párhuzamos korpusz tesztelésre elkülönített részénél összevetettük az OCR és *Silver* anyagot. Az anyagok közötti különbségeket listáztuk, gyakoriság szerint rendeztük. A gyakorisági listából látható volt, hogy a különbségek jelentős része nem OCR-eredetű hiba, vagy az OCR-hiba javítása szempontjából nem releváns, a kiértékeléskor nem veendő figyelembe. Példák:

– A könyvek oldalszámozása, mely az OCR-szövegben megvolt, a *Silverből* azonban hiányzott. Ez a kiadások közötti különbség, nyilvánvalóan nem OCR eredetű hiba. Ezeket a hibákat koherenciahibaként (K) címkéztük fel. Koherenciahiba származhat az előfeldolgozásból is: előfordult, hogy OCR tévesztés miatt a mondattokenizálás eltért az OCR és a *Silver* szövegekben, ami rontotta a párhuzamosítást.
– A különbségek másik része az eltérő kiadások közötti központozásból származott. Ezeket punktuációs (P) eltérésként címkéztük fel. Előfordul például az OCR és a *Silver* korpusz között a *vessző* (,) és a *pontosvessző* (;) karakterek felcserélődése. Ennek javítását nem tekintjük OCR-hiba-javításnak, hiszen a szöveg minőségét nem javítaná, valamint könnyen lehet, hogy ezek is kiadásbeli különbségből származnak.

Ily módon rendszerezve a hibákat elértük, hogy a *silver standarden* történő kiértékeléskor csak az OCR-hibákon legyenek mérve a modellek teljesítménye.

A 2. táblázatban látható néhány példa a fent vázolt OCR-hiba címkézési stratégiájára.

A különbségek megtalálásakor a Pythonhoz írt *Levenshtein*[10] függvénytárát használtuk.

Miután a hibák felcímkézését ily módon elvégeztük, a címkéket karakter alapon rendeltük hozzá az OCR-*silver* párokhoz. Ehhez szükség volt arra, hogy a párhuzamos korpuszt mondatonként azonos hosszúságúra kitöltsük, majd a mondat párokhoz illesszük a hibaosztályokat. A CCC detekciós modell finomhangolásakor ugyancsak azonos hosszúságúra hozott mondatpárokat vár a bemenetre:

```
[OCR_toInput]  mi rést nappal törtek, azt éjjel újra megépítteté s a vár 196 ellen fúrt
[OCR_aligned]  mi rést nappal törtek, azt éjjel újra megépítteтнs a vár 196 ellen fúrt
[ GS_aligned] Ami rést nappal törtek, azt éjjel újra megépítteté,s a várнннн ellen fúrt
[OCRGD_errcl] O                                                P     KKKK
```

1. ábra: Azonos hosszúságúra töltött OCR-*silver*-hiba osztály címkesor. Az 'O' az OCR, a 'P' a központozási-, a K a koherencia hiba címkéje.

[10] https://pypi.org/project/python-Levenshtein

OCR	Gold	Hibaosztály	Előfordulás	Megjegyzés
é	e	ocr	581	karaktercserélődés
zd	v	ocr	1	több karakterből kevesebb
űié	üle	ocr	19	együtt többször előforduló összetett eset
szóköz	¤	ocr	3523	szóösszeragadás
¤	szóköz	ocr	360	szószétesés
y	¤	ocr	11	karakterelnyelődés
¤	r	ocr	232	karakterfelbukkanás
5	¤	koherencia	7	oldalszám az ocr-ben
-5	¤	koherencia	7	elválasztás és oldalszám az ocr-ben
ioo	¤	koherencia	4	hibásan ocr-ezett oldalszám
¤	(Hej	koherencia	1	előfeldolgozási hiba
:	,	központozási	123	eltérő kiadásból származó
,	;	központozási	50	eltérő kiadásból származó

2. táblázat. Példák OCR-silver között előforduló hibákra és osztályozásukra a JiM teszt párhuzamos korpuszában. A ¤ karakter a karakterhiányt szimbolizálja.

3.3. *Gold standard* korpusz előállítása

A párhuzamos korpuszt további manuális javításnak vetettük alá, hogy előállítsuk a *Gold standard* korpuszunkat. A különböző kiadások miatti eltéréseket négy annotátor kézzel javítja.[11] A javításhoz rendelkezésükre áll a – hibákat tartalmazó – OCR kimenet és az eltérő kiadásból származó hibátlan digitalizált szöveg (azaz a *Silver standard* korpusz). Az eredeti pdf-et felhasználva igazítják (azaz módosítják) a digitalizált változatot ahhoz a kiadáshoz, amiből az OCR kimenete készült. A végső, kézzel javított elektronikus korpusz a *Gold standard* korpuszunk. A javítás főbb alapelvei a következők:

– A javítás során nem a pdf fájlot követjük, hanem az OCR-ezett szöveget és a *Silvert*. A fájlok ugyan követik a pdf-et, de akár egész bekezdések is hiányozhatnak belőlük a pdf-hez képest. Éppen ezért csak az OCR-ezett szöveg és a *Silver* közti eltérő szövegrészek kontextusára keresünk rá a pdf-ben, és azt javítjuk.
– A helyesírást ignoráljuk: a digitalizált változatban szereplő *feleletet* szót ki kell javítani *feletet*-re, ha a pdf-ben el van írva, függetlenül attól, hogy sajtóhiba.
– Ha az OCR-ben van egy mondat, ami a digitalizált változatból hiányzik, de a pdf-ben megtalálható, akkor azt beírtuk a digitalizált változatba, oda, ahol a OCR-ezett változatban található.
– Két dolgot nem veszünk át a pdf-ből a digitalizált változatba: Ha egy szóban elválasztás van az OCR-ben, és tényleg el van választva a pdf-ben is, mert sortörés van benne, azt nem írjuk elválasztással a javított szövegben (tehát a pdf-ben *Magyaror-ság* van, az OCR-ezett szövegben *Magyaror-ság*,

[11] A cikk megjelenésének időpontjában 267 448 mondat ellenőrzése készült el (3 515 401 szó).

a digitalizált változatban *Magyarország*; ez utóbbit így is hagyjuk). Ha az oldalszám szerepel az OCR-ezett változatban és szerepel a pdf-ben is, akkor sem írjuk be a digitalizált változatba.
– Az OCR-ezett szövegben gyakran extra szóköz szerepel az írásjelek előtt. Hiába lehet a pdf-ben szellősen szedve a szöveg, a *Gold standard* változatban tapadnia kell az írásjelnek a szóvéghez.

Kutatásunk jelen a fázisában még csak a *Silver standard* korpusszal történő kiértékelések láthatóak.

4. OCR-hibát detektáló és javító módszerek

Az OCR-javításhoz 4 különböző módszerrel kísérleteztünk. Kipróbáltunk egy detekciós-, illetve két különböző javítómódszert.

Kutatásunk során magyarra finomhangoltuk és implementáltuk a **CCC** többnyelvű BERT alapú detektáló módszerét, amely a 2019-es OCR-hibajavító verseny (Rigaud és mtsai, 2019) győztesének módszere. Betanítottunk továbbá egy transzformer alapú neurális gépifordító rendszert, és végül finomhangoltunk egy előtanított BART modellt a javítófeladat megoldásához.

A **CCC** (Context-based Character Correction, Rigaud és mtsai, 2019) módszer detekciós modulja (lásd 2. ábra) 3 fő részből áll: BERT enkódoló, konvolúciós háló, bináris osztályozó.

A BERT enkódoló segítségével tudja a modell megtanulni a környezetfüggő tulajdonságokat az adott szövegről, majd a BERT modellel enkódolt adatot 4 különböző kernellel rendelkező konvolúciós rétegekbe küldi be. A különböző konvolúciós rétegek gyakorlatilag n-gramm információkat osztanak meg egymással. Végül a konvolúciós rétegek által kiadott adatokat konkatenálva továbbadja egy teljesen összekapcsolt bináris osztályzást végző rétegnek, így létrehozva a token szintű értékelőt.

A CCC módszer kódja nem publikus, de egy TDS cikk[12] alapján reprodukáltuk, majd integráltuk a huBERT (Nemeskey, 2021) modellt, hogy magyar nyelvre tudjon detektálni. A finomhangoláshoz szükséges párhuzamos korpuszt a JiM nyers változata szolgáltatta, míg a kiértékelés a JiM *silver* verzióján történt. A modell paramétereit módosítottuk úgy, hogy 250 szubtoken-t is képes legyen kezelni. Emellett kipróbáltunk több verziót (tanítottunk bert-base-multilingual-cased modellt, illetve az ICDAR2019-es több nyelvű tanító anyaggal is tanítottuk a modelleket), de az előzetes várakozásnak megfelelően a huBERT + JiM korpusz szolgáltatta a legjobb kiértékelési eredményeket.

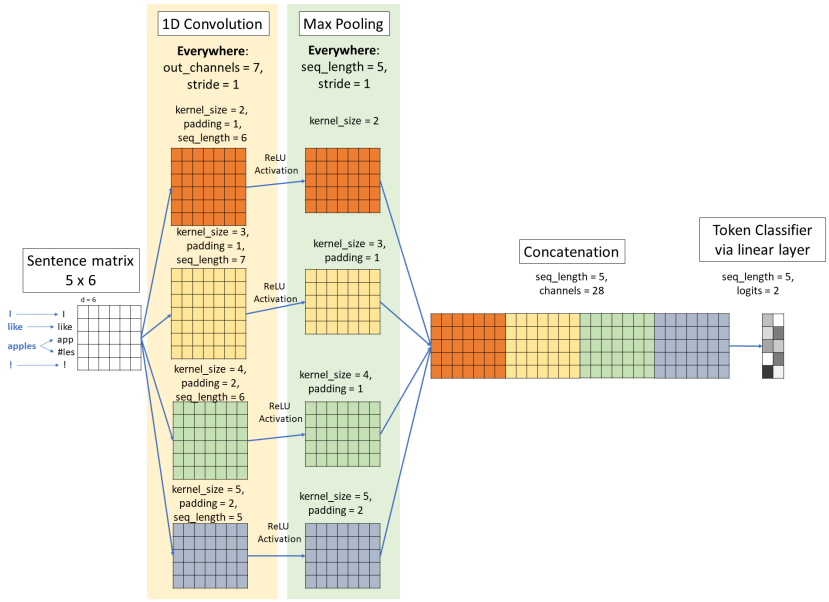
Megemlítendő még, hogy a detektáló *gold* anyagon vélhetően még jobban fog teljesíteni, mint *silveren*, mivel a koherenciát megbontó, oda nem illő szövegrészek láthatóan megzavarták a detektor viselkedését. Példa:

OCR eredeti mondat: *A csodabúvár szót fogadott tündér anyjának, s vágva* ***egy 45 jó birssuhogót*** *az erdőről, beveté magát a vízbe a sárga liliomokon túl.*

[12] https://towardsdatascience.com/ocr-typo-detection-9dd6e396ecac

A vastag betűs szavakat tartja a detektáló gyanúsnak, ami -legalábbis az **egy** és a **jó** szavak esetében - nyilván a közbeékelődő oldalszámnak (**45**) köszönhető.

Az osztályozó elvi felépítése:



2. ábra: CCC detekciós modell architektúrája

A **Marian NMT** (Junczys-Dowmunt és mtsai, 2018) egy neurális gépifordító keretrendszer, amely C++ nyelven íródott és szabadon hozzáférhető. Könnyen telepíthető, jól dokumentált, memória- és erőforrás-optimális implementációjának köszönhetően[13] az akadémiai felhasználók és fejlesztők által leggyakrabban használt eszköz (Barrault és mtsai, 2019). A Marian NMT egy figyelmi (attention) modellel támogatott enkóder-dekóder architektúrájú neurális gépifordító modell, amely támogatja a transzformer architektúrát is. A legnagyobb előnye, hogy a többi módszerhez képest a leggyorsabb tanítás érhető el és ehhez nincs szükség előtanított nyelvmodellekre. A betanított modell paraméterei: 6 réteg enkóder és 6 réteg dekóder; 8 figyelmi fej; 512 szóbeágyazás dimenzió; bementi hossz: 512; előre csatolt háló méret: 2048.

A probléma egy gépi fordítás feladatként is értelmezhető, ahol a forrásnyelv egy hibás OCR-szöveg, a célnyelv a hibás OCR-szöveg hibátlan párja. A természetes nyelvek közötti gépi fordítás mondatszinten történik, ahol mind a tanítóanyagnak, mind a fordítandó szövegnek mondatokra bontottnak kell lennie. Az OCR-javítás feladatnál ezzel szemben nagyobb egységek vannak, és a dokumen-

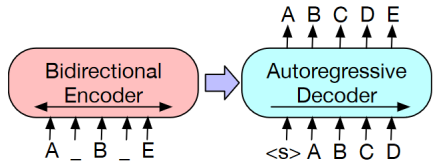
[13] https://marian-nmt.github.io

tum nincsen mondatokra bontva. Kézenfekvő megoldás, hogy fordítási egységnek a 100-as ablakra tagolt szöveget használjuk. Azonban a méréseink szerint a nagyon rövid szegmensek esetén a rendszer nem adott értelmezhető kimenetet. Ennek a jelenségnek a kezelése érdekében a végleges rendszert a 100-as ablakra és a mondatokra bontott szövegen vegyesen tanítottuk, így alkalmassá vált tetszőleges méretű szövegek javítására.

A **BART** (Lewis és mtsai, 2020) modell egy enkóder-dekóder architektúrán alapuló transzformer modell (lásd 3. ábra), amelyet a Facebook fejlesztett[14]. Az enkóder kétirányú (Bidirectional), a dekóder autoregresszív (Autoregressive). A BART gyakorlatilag ötvöz egy BERT (Devlin és mtsai, 2019b) és egy GPT (Radford és Narasimhan, 2018) típusú modellt. A kutatások alapján a BERT típusú modellek alkalmasak arra, hogy jó reprezentációt készítsenek egy adott szövegről, azonban szöveggenerálási feladatokra kevésbé megfelelőek, míg a GPT típusú autoregresszív modellek elsősorban szövegek generálására működnek jól. A BART a két architektúra előnyeit ötvözi, ezért kiválóan alkalmas például gépi fordításra. A BART rendszert két különböző méretben tették közzé:

– BART-base: 6 réteg enkóder és 6 réteg dekóder; 12 figyelmi fej; 768 szóbeágyazás dimenzió; bementi hossz: 512; 140 millió paraméter
– BART-large: 12 réteg enkóder és 12 réteg dekóder; 16 figyelmi fej; 1024 szóbeágyazás dimenzió; 1024 bemeneti hossz; 400 millió paraméter

A BART modell finomhangolható gépi fordításra, előnye a Marian NMT-vel szemben, hogy előtanított dekóderrel indítja a finomhangoló tanítást. A kutatásunkban egy magyar nyelvű BART-base modellt finomhangoltunk.



3. ábra: BART modell architektúrája (Lewis és mtsai, 2020)

5. OCR-hibajavító kísérletek

Kutatásunk során négy különböző javítási kísérletet végeztünk az OCR-hibák javítására:

– Marian NMT rendszerrel betanított gépifordító
– Magyar nyelvű BART finomhangolása gépifordító rendszerre
– CCC detekció és Marian NMT rendszerrel betanított gépifordító

[14] https://github.com/pytorch/fairseq/tree/master/examples/bart

– CCC detekció és Magyar nyelvű BART finomhangolása gépifordító rendszerre

Az OCR-hibajavító neurális hálózatok használatának általános mellékhatása, hogy sokszor kijavítanak olyan szövegrészeket is, amelyekben nem volt hiba (False Positive módosítás). Ezért – követve az utóbbi évek trendjeit – integráltuk a CCC detekció módszerét, hogy szűkítse le a helyeket, ahol vélhetően szükséges a javító alkalmazása.

Mivel az ICDAR versenyeken több európai nyelv is szerepelt a tanítóanyagban, a CCC detekciós modellje egy *bert-base-multilingual-cased* modellt vett alapul. Ezt – lévén a JiM magyar nyelvű – kicseréltük egy *huBERT-base-cc* modellre. Ezt tanítottuk be a JiM tanító adathalmazával, amelyet e célból az ICDAR megmérettetés sztenderd formátumára hoztunk. A CCC modell az osztályozást tokenszinten végezte el, ami ez esetben szavak, illetve központozó elemek megjelölését tette lehetővé. Nem tudta jelölni azonban a szóközöket, azaz a szószétéseséseket ez a módszer nem tudja detektálni. Éppen ezért a szóközöket minden esetben potenciálisan hibásnak feltételeztük akkor, amikor a detektorral a javítók módosítási javaslatait megszűrtük.

A Marian NMT tanításához és a BART finomhangolásához konkatenáltuk a mondatalapú és a 100 karakteres tanító korpuszokat, majd megkevertük a sorokat. A BART esetében további tokenizálást is végrehajtottunk az Orosz Györgyféle magyar nyelvű spaCy modellel (0.3.1).[15]

A BART finomhangolásához 4 darab GeForce GTX 1080Ti GPU kártyát használtunk, az alábbi módosított hiperparaméterekkel: batch méret / GPU: 14; epoch: 30; maximum bemeneti és kimeneti szöveg hossz: 256; fp16; tanulási ráta: $2e^{-8}$; szótár méret: 40 000; warmup: 15 000. A BART esetében az epoch szám fixen 30.

A Marian NMT tanításához 2 darab GeForce GTX 1080Ti GPU kártyát használtunk, az alábbi módosított hiperparaméterekkel: epoch: 34; maximum bemeneti szöveg hossz: 100; tanulási ráta: $3e^{-4}$; spm (Kudo és Richardson, 2018) szótár méret: 32 000; warmup: 16 000. A Marian NMT esetében az epoch szám a validációs halmazon való kiértékeléshez kötött, a megállási feltétel (early stopping) 10-re volt beállítva, ami 34 epoch számnál állította le a tanítást.

A CCC detekciós modell tanításához 4 darab GeForce GTX 1080Ti GPU kártyát használtunk, az alábbi hiperparaméterekkel: epoch: 3; maximum bemeneti szöveg hossz: 512; tanulási ráta: $3e^{-5}$; warmup: 0.

6. Eredmények és kiértékelés

A 3. és a 4. táblázatban láthatóak az OCR-javító módszerek eredményei. A kiértékeléshez az alábbi metrikákat használtuk (Nguyen és mtsai, 2021): abszolút pontosság (*accuracy*), relatív pontosság (*precision*), fedés (*recall*), F-mérték (*F1-score*) és karakterszintű hibaarány (*character error rate* – CER). A kiértékelésben kétféle megközelítést választottunk. Az első a csak OCR-hibák vizsgálata,

[15] https://github.com/spacy-hu/spacy-hungarian-models

a másodikban az OCR-hibák mellett a koherencia- és központozási hibákat is belevettük a kiértékelésbe, mint például írásjelek konzisztenciája, vagy oldalszámok beékelődése a szövegbe stb. Az eredmények azt mutatták, hogy a fordítók ezeket a nem OCR-típusú hibákat is tudták detektálni, javítani.

A 3. táblázatban láthatóak a csak OCR-hibák javításában elért teljesítmények. Az eredményben az látható, hogy a BART és a Marian NMT modellek tekintetében a BART magasabb fedéssel, míg a Marian NMT magasabb pontossággal javít, ezek mértéke határozta meg a végső F-mértékek eredményét, ahol a CCC nélküli versenyben a Mariann NMT győzőtt, míg a CCC integráció esetében a BART jött ki győztesen (F-értékben mérve). Ez azt jelenti, hogy amit a Marian NMT javít, azt pontosan javítja, azonban kevesebbet javít. Ezzel szemben a BART többet javít, ezáltal magasabb a fedése, ami nagyobb pontatlansághoz is vezet. Továbbá az látható még, hogy a CCC integrációja tovább növeli az összes modell teljesítményét.

	Pontosság	Fedés	F-mérték
BART	46,96	**42,61**	44,67
Marian NMT	53,08	42,11	46,95
CCC + BART	64,82	40,43	**49,80**
CCC + Marian NMT	**65,33**	40,09	49,69

3. táblázat. Modelljeink eredményei az OCR-hibák javításában

A 4. táblázatban láthatóak az OCR- és koherenciahibák javításának eredményei. Az eredményekből látszik, hogy a BART és a Marian NMT magas minőségben képesek a koherenciahibákat is javítani; mivel nagyobb szövegegységeken lettek tanítva, ezáltal a mondatszintű hibákat is képesek kezelni. Ebben a feladatban a Marian NMT érte el a legjobb eredményt, ami azt jelenti, hogy a Marian NMT jobban meg tudta tanulni a szövegszintű koherenciahibákat. A 4. táblázatban nem látható a CCC integráció, mivel a CCC csak OCR-hibákat tud detektálni.

	Pontosság	Fedés	F-mérték	Accuracy	1-CER
BART	59,78	70,25	64,59	99,35	0,9959
Marian NMT	**73,39**	**70,54**	**71,94**	**99,53**	**0,9969**

4. táblázat. Modelljeink eredményei az OCR- és koherenciahibák javításában

7. Összegzés

Jelen cikkben napjaink kurrens kihívására, a OCR-hibajavítás problémára szolgáltattunk több, mélytanuláson alapuló megközelítést, speciálisan magyar nyelvű szövegek javítására. Az irodalomkutatás, valamint a tárggyal kapcsolatos nemzetközi megmérettetés eredményei azt mutatták, hogy a problémára adott *state-of-the-art* megoldások kétlépcsősek, detektor és javító egységek egymásra épüléséből állnak. Implementálásra a változatos detektálási lehetőségek közül a BERT-alapút választottuk, míg a javítást gépi fordítási feladatként értelmeztük.

Mindkét alegységhez, az egységek betanításához szükség volt tanítóanyagra, azaz megfelelő méretű párhuzamos korpuszra, ez azonban a projekt kezdetén nem állt rendelkezésre, annak megteremtése is a feladat része lett. Végül olyan párhuzamos korpuszt sikerült létrehoznunk, mely megfelel a rövid és hosszabb távú – a cikk tárgyán túlmutató – céljainknak:

- A párhuzamos korpusz egyik fele az Arcanum OCR-ezett szövegeiből
- másik fele az ennek megfelelő, OCR-hibák nélküli elektronikus alakból álljon.
- A párhuzamos korpusz kellően nagy méretű legyen a neurális modellek tanításához.
- A korpusz előfeldolgozása rövid időn belül elvégezhető legyen.

E szempontok alapján sikerült Jókai Mór és Mikszáth Kálmán életművének elemeiből előbb *Silver standard*, majd *Gold Standard* anyagot létrehozni.

A betanítás és kiértékelés lépéseit úgy terveztük meg, hogy a fordítómodulok több előnyös tulajdonságát is be tudjuk mutatni, nem pusztán a szűken vett OCR-hiba javításának minőségét. Eredményeink azt mutatják, hogy az általunk létrehozott architektúra nem csak OCR-hibákat tud javítani, hanem koherenciaproblémákat is tud kezelni, mint például az oldalszámok automatikus negligálása, vagy az elválasztott szavak egyesítése.

Köszönetnyilvánítás

Ezúton is szeretnénk megköszönni az Arcanum Adatbázis Kft.-nek, hogy rendelkezésünkre bocsátották erőforrásaikat, és ezzel lehetővé tették a hibajavító rendszerek fejlesztését.

Hivatkozások

Barrault, L., Bojar, O., Costa-jussà, M.R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., MĂźller, M., Pal, S., Post, M., Zampieri, M.: Findings of the 2019 conference on machine translation (wmt19). In: Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1). pp. 1–61. Association for Computational Linguistics, Florence, Italy (August 2019)

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 2019a)

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 2019b)

Duong, Q., Hämäläinen, M., Hengchen, S.: An unsupervised method for OCR post-correction and spelling normalisation for Finnish. In: Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa). pp. 240–248. Linköping University Electronic Press, Sweden, Reykjavik, Iceland (Online) (May 31–2 Jun 2021)

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A.F.T., Birch, A.: Marian: Fast neural machine translation in C++. In: Proceedings of ACL 2018, System Demonstrations. pp. 116–121. Association for Computational Linguistics, Melbourne, Australia (July 2018)

Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.: OpenNMT: Open-source toolkit for neural machine translation. In: Proceedings of ACL 2017, System Demonstrations. pp. 67–72. Association for Computational Linguistics, Vancouver, Canada (Jul 2017), https://aclanthology.org/P17-4012

Kudo, T., Richardson, J.: SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 66–71. Association for Computational Linguistics, Brussels, Belgium (Nov 2018), https://www.aclweb.org/anthology/D18-2012

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7871–7880. Association for Computational Linguistics, Online (Jul 2020)

Mei, J., Islam, A., Wu, Y., Moh'd, A., Milios, E.E.: Statistical learning for OCR text correction. CoRR abs/1611.06950 (2016), http://arxiv.org/abs/1611.06950

Nemeskey, D.M.: Introducing huBERT. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia. pp. 3–14. Szegedi Tudományegyetem, Informatikai Intézet, Szeged, Magyarország (2021)

Nguyen, T.T.H., Jatowt, A., Coustaty, M., Doucet, A.: Survey of post-ocr processing approaches. ACM Computing Surveys 1, 1 (March 2020), 36 (Mar 2021), https://doi.org/10.5281/zenodo.4640070

Nguyen, T.T.H., Jatowt, A., Nguyen, N.V., Coustaty, M., Doucet, A.: Neural machine translation with bert for post-ocr error detection and correction. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020. p. 333–336. JCDL '20, Association for Computing Machinery, New York, NY, USA (2020)

Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training (2018)

Rigaud, C., Doucet, A., Coustaty, M., Moreux, J.P.: Icdar 2019 competition on post-ocr text correction. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1588–1593 (2019)

Schaefer, R., Neudecker, C.: A two-step approach for automatic OCR post-correction. In: Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature. pp. 52–57. International Committee on Computational Linguistics, Online (Dec 2020)

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 30, pp. 5998–6008. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf