OCR-hibák kvantitatív elemzése több szövegváltozat összehasonlításával

Pethő Gergely, Sass Bálint, Simon László, Lipp Veronika

HUN-REN Nyelvtudományi Kutatóközpont, Lexikológiai Intézet
pagstudium@gmail.com, {sass.balint,simon.laszlo,lipp.veronika}@nytud.hun-ren.hu

Kivonat Az optikai karakterfelismeréssel (OCR) feldolgozott digitalizált dokumentumokba különböző okokból (szokatlan betűtípus, szennyezett oldal, nem tökéletes nyomtatás stb.) hibák kerülnek, amelyek rontják a dokumentum olvashatóságát és további használhatóságát például korpusz építése vagy nyelvmodell tanítása tekintetében. Ahhoz, hogy ezeket javítani tudjuk, hasznos ismernünk, hogy az OCR-alkalmazások milyen betűket és betűkombinációkat milyen gyakran mire szoktak rontani. Az általunk bemutatott eljárással nagy mennyiségű magyar nyelvű szkennelt oldalkép több OCR-programmal felismert szövegváltozatát összehasonlítva nyertünk adatokat az OCR-hibák gyakoriságáról és az alkalmazások relatív pontosságáról. Annak eldöntését, hogy az eltérő változatok közül melyik a helyes, illetve rontott, karakterszintű nyelvmodellre bíztuk. Cikkünkben közöljük a módszerünk leírását, a magyar OCR-hibák statisztikáját, valamint az egyes alkalmazások hibaarányait.
Kulcsszavak: OCR, optikai karakterfelismerés, karakterszintű nyelvmodell, rekurrens neurális hálózat, karakterfelismerési hibák

1. Bevezetés

Rice és mtsai (1999) szerint az optikai karakterfelismeréssel (OCR) nyert digitalizált szövegekben megjelenő hibák három különböző okra vezethetőek vissza, amelyek hasonló arányban felelősek a problémákért: 1. képhiba, 2. karaktertévesztés (a hasonló karakterek téves azonosítása, különös tekintettel a központozás írásjeleire), 3. tipográfiai nehézségek (stílus, dőlt, kövér szedés, térköz, nagyon kis vagy nagy méretű betűk stb.).

Az OCR a tapasztalatok szerint betűről betűre haladva, a kontextust ignorálva nem működik, mivel (1) sokszor egy-egy betű határainak a megállapítása is nehézségekbe ütközik; (2) ha a betű képe az emberi szem számára egyértelmű is, akkor is több betűnek vagy betűsorozatnak lehet megfeleltethető. Ezért kulcsfontosságú a nyelvi kontextus: az OCR-alkalmazás sokszor csak ez alapján képes helyesen dönteni arról, hogy a betűkép valójában más betűnek feleltethető meg, mint amelyhez a leginkább hasonlít.

Az OCR-rel előállított digitális állományokból létrehozott korpusz minősége jelentősen különbözhet pl. a nyomtatás minőségétől és a használt betűtípusoktól függően. A bennük előforduló zaj aránya komoly negatív hatással lehet a későbbi szövegbányászati eljárások pontosságára, vö. D'hondt és mtsai (2017).

Cikkünkben egy olyan kimutatás elkészítéséről számolunk be, amelyből kiderül, hogy az OCR-alkalmazások milyen tévesztésekre „hajlamosak", valamint közreadjuk a hibák gyakorisági listáit is. Úgy véljük, ez számos téren hasznos lehet az OCR-hibák mennyiségének csökkentése tekintetében: szerepet játszhat akár egy OCR-javító alkalmazás konfigurálásában, akár olyan nyelvmodell fejlesztésében, amely pl. zajszűrő autoencodert alkalmazva célzottan megtanulja felismerni és kezelni a jellegzetes OCR-hibákat.

2. Irodalmi előzmények

Bassil és Alwani (2012) az OCR-hibák javítására három algoritmus kombinációját javasolta: az egyik a hiba felismerését, a másik a helyes szóalak generálását, a harmadik a hibajavítást végzi, és mindhárom algoritmus a Google Web 1T 5-gram adatkészletéből nyert információkat használja fel. Meglátásuk szerint kétféle „helyesírási" hiba létezik: egyrészt az invalid tokent eredményező hiba, másrészt az az eset, amikor a karaktertévesztés helyes szóalakot hoz létre, amely azonban szemantikailag nem illeszkedik a szövegkörnyezetbe.

Kissos és Dershowitz (2016) szerint az OCR-ezés során keletkező hibák eloszlása nem csupán a kép minőségétől függ, hanem a felismerés tárgyának nyelvétől is. Úgy vélik, hogy a hibák három fő típusba sorolhatóak be: 1. a képen lévő szöveg felismerésének sikertelensége miatt az OCR nem tudja beazonosítani a helyes szóalakot; 2. a szóközök fel nem ismerése miatt a szavak szegmentálása hibás; 3. az egyes karakterek szegmentálása hibás. A korrekciót célzó módszertanuk három szakaszból áll. Az első lépésben egy tévesztési mátrix és egy szótári keresés segítségével megállapítják a hibás szó helyett potenciálisan szóba jöhető helyes szóalakokat, majd azokhoz egy vektort rendelnek. Ezt a vektor minden egyes szavához tartozó jellemzők kinyerése követi, majd egy kétlépcsős osztályozás történik, ahol (1) az adott szóalak adott pozícióban való helyességének valószínűsége szerinti rangsorolás után (2) a legvalószínűbb szó kiválasztása következik.

D'hondt és mtsai (2017) tanulmányukban egy új megközelítésről adnak számot, amely nem igényel terjedelmes tanítókorpuszokat. A szövegbeli hibákat felismerő rendszerüknek csak kis mennyiségű, ám viszonylag tiszta és reprezentatív korpuszból származó tanítóadatra van szüksége ahhoz, hogy megtanuljon egy karakteralapú modellt, amely BiLSTM-en alapul. A szerzők arra a következtetésre jutnak, hogy a hagyományos hibafelismerő rendszerek a valószínűsíthető karakterek tévesztési mátrixainak megkonstruálására összpontosítottak, azaz a szavak elrontásának felismerésére. A legújabb rendszerek azonban a hiba megjelenésének nyelvi kontextusára vonatkozó információk figyelembevételével javítják a pontosságot, amihez többek között az internetről származó nagy mennyiségű szó- és karakterszintű n-gramokat használnak fel.

Kamlah és Stegmüller (2018) annak lehetőségét vizsgálta, hogy több, egymástól eltérően hibás OCR összevetésével lehet-e az inputok bármelyikénél jobb outputot kapni. Az eredményeik részben biztatóak, ugyanakkor eljárásuk alkalmazhatóságát nagyban korlátozza, hogy ahhoz olyan OCR-outputra van szükség, amely karakterenként megadja a felismerés konfidenciaértékét. A magyarra ezt

a funkcionalitást egyedül a nagyvállalati vásárlók számára elérhető FineReader Server kínálja.

3. Módszer

Módszerünk abból a megfigyelésből indul ki, hogy ha több OCR-alkalmazással, vagy egyetlen ilyennel, de a feldolgozási folyamaton változtatva (például a szkennelt kép metaadataiban a felbontás értékét kissé módosítva) dolgoztatjuk fel **ugyanazt** az oldalképet, a kimenetként kapott szövegváltozatok számos ponton eltérnek egymástól. Igaz, ezen eltérések jelentős része nehezen felismerhető helyeken jelentkezik, amelyeket egyik alkalmazás sem tud sikeresen feldolgozni, és így minden kapott kimenet lényegében véletlen zaj lesz, ami éppen véletlen jellege miatt tér el a változatok között. Szintén előfordul, hogy egy jól olvasható részt valamilyen érthető okból (pl. átüt egy festékfolt a lap túloldaláról, ami helyzete alapján hasonlít egy ékezetre) minden alkalmazás tévesen ismer fel. Ugyanakkor e számunkra érdektelen, hasznosíthatatlan esetek mellett gyakori az is, hogy egy nem tökéletesen olvasható betűt egy vagy több alkalmazás eltéveszt – azonos vagy eltérő módon –, míg egy másik alkalmazás azt sikeresen felismeri.

Megközelítésünk lényege, hogy nagy mennyiségű valódi szkennelt folyóirat- és újságoldalképet több OCR-alkalmazással dolgoztatunk fel. Azok kimenetei között lokálisan, a bemenetként szolgáló oldalképen elfoglalt helyük alapján azonosítunk pontszerű eltéréseket (pl. egyik dokumentumban *ü* szerepel, ahol a másikban *ii*). Az eltérésből következik, hogy nem lehet mindegyik felismerési változat helyes. Ha nem is lehetünk biztosak abban, hogy az alternatívák között ott van a nyomtatott oldalképnek megfelelő változat, abból kiindulhatunk, hogy normális esetben rendszerint ott lesz. A „nem normális" esetek lényege, hogy az adott régióban valamilyen mintázat (pl. szokatlan formájú betű, mintás háttér, nyomtatási vagy szkennelési hiba stb.) látható, de az nem hasonlít eléggé egyetlen szokványos betűhöz sem. Így mindegyik alkalmazás elrontja ezeket, de mind másképp, azaz mind eltérő karaktersorozatot „lát bele".

Mindebből következik, hogy ha összegyűjtjük sok szövegben az OCR-változatok közötti pontszerű eltéréseket, kétféle esetet fogunk találni:

- **szisztematikus** tévesztéseket, ahol az alternatívák között többnyire **szerepel** a helyes változat, illetve
- a szövegbe beszűrődő kosz, háttér, fényképrészlet stb. által kiváltott nem szisztematikus, **egyedi, véletlenszerű** hibákat, amelyek között rendszerint **nem szerepel** a helyes szövegváltozat, vagy nincs is felismerendő szöveg.

A dokumentumok közötti pontszerű eltéréseket alkotó változatpárok összeszámolásával átfogó és pontos számokkal alátámasztott áttekintést kapunk arról, hogy melyek a további alkalmazások – pl. OCR-szövegek automatikus javítása – szempontjából érdekes, mivel **rendszeresen visszatérő**, és melyek szórványos hibatípusok. Ha még azt is megbízhatóan és automatizáltan el tudjuk dönteni, hogy a változatok közül melyik a helyes és melyik a rontott az adott kontextusban, akkor egyrészt a tévesztések **irányáról** is pontos képet tudunk alkotni (például

az *ii* és *ü* közül majdnem mindig az *ü* a helyes), valamint az összehasonlított **OCR-szoftverek hibázási arányát** is mérni tudjuk, konkrétabban azt, hogy az eltérések mekkora hányadában téved az egyik, illetve a másik alkalmazás. Az utóbbi célra egy szabadon elérhető karakterszintű rekurrens nyelvmodellt használunk, amellyel megmérjük az alkalmazások által javasolt változatok perplexitását az adott kontextusban. Azt tekintjük helyesnek, amely a nyelvmodell szerint alacsonyabb perplexitású. A nyelvmodell predikcióit kézi ellenőrzéssel validáltuk.

3.1. Anyagok

Szövegek A feldolgozás tárgyát 78 magyar nyelvű sajtótermék (napi- vagy hetilap, magazin, folyóirat) 2001 és 2022 között megjelent számai alkották 1 millió oldal terjedelemben. A legnagyobb oldalszámokkal az Új Dunántúli Napló (120 000), a Békés Megyei Hírlap (90 000), a Délmagyarország, a Délvilág és a Családi Kör című vajdasági lap (75-75 000) képviseltette magát, a többi periodikum anyaga néhány ezertől néhány tízezerig terjedt.

OCR-alkalmazások A sajtótermékek egyszer szkennelt, JPEG formátumú oldalképeit a rendelkezésünkre álló FineReader 14-es, 15-ös és 16-os verziójával (a továbbiakban FR14, FR15, illetve FR16) ismertettük fel. Más alternatíva nem kínálkozott, miután az egyetlen szabad OCR-program, a Tesseract magyar nyelvű modelljét rövid tesztelés után használhatatlannak találtuk: még jó minőségű, tiszta oldalképeken is gyenge a felismerési pontossága. Ezeken kívül csak software-as-a-service alapú, oldalanként fizetős megoldások jöhetnének szóba, amelyeket eleve kizártunk.

A három alkalmazást egy-egy alsó középkategóriás asztali számítógépen futtattuk gyári alapbeállításokkal Windows alatt, szkriptelve, az alkalmazások parancssori interfészét használva, magyar felismerési nyelvvel. A feldolgozás végeredményét PDF-fájlokba mentettük. A gépek egy bő hónapig folyamatosan futva állították elő az egyenként 1 millió oldalnyi PDF-dokumentumot. A három szoftver azonos oldalakon eltérő számú sort azonosított: 190,8 (FR14), 223,5 (FR15), illetve 221,2 millió (FR16). A felismert szövegszavak száma ugyanakkor mindhárom esetben kb. 900 millió volt.

3.2. Feldolgozási lépések

Sorok kinyerése a PDF-dokumentumokból A PDF formátum összefüggő szöveget még sorok vagy szavak szintjén sem tartalmaz, a betűket külön-külön, a lapon elfoglalt koordinátáik alapján jellemzi. Mivel a PDF-ben található szöveget nem karakterek, hanem sorok szintjén akartuk feldolgozni, e fájlok tartalmát át kellett alakítanunk. A PDF lineáris szöveggé alakítására különböző PDF-olvasók és -szerkesztőprogramok állnak rendelkezésre. Mi ehhez a PyMuPDF Python-csomagot használtuk, az általa talált sorok szövegét és az azokat egyenként körbezáró téglalapok koordinátáit JSON formátumban tároltuk.

Sorok illesztése A kapott JSON-fájlokat páronként összehasonlítottuk – az FR14 outputjából származó változatokat az FR15-éivel, valamint az FR15-öt az FR16-tal –, és oldalról oldalra megkerestük az egymáshoz legközelebb eső sorokat. Ehhez kiszámítottuk a sorok mértani középpontjait, majd az összes lehetséges sorpár középpontjai közötti síkbeli (euklideszi) távolság és szövegük Levenstejn-távolsága alapján a két OCR-változat sorai között távolságmátrixot alkottunk. Mohó eljárással lépésenként az egymáshoz legközelebbi sorokat egymáshoz tartozónak tekintettük, és eltávolítottuk a mátrixból. Ennek során felső távolságkorlátot is alkalmaztunk, hogy sem egymástól nagyon eltérő szövegű, sem térben nagyon távol eső sorokat ne illesszünk egymáshoz. A végén pár nélkül maradt sorokat megpróbáltuk 2:1-hez illeszteni oly módon, hogy kezelni tudjunk olyan eseteket, amikor az egyik alkalmazás két vízszintesen egymás mellett elhelyezkedő sort ismert fel, míg a másik azt egyetlen sornak látta.

Eltérések azonosítása Az illesztett sorpárokat a Python `difflib` könyvtárának `SequenceMatcher` osztályával, `autojunk=False` paraméterrel hasonlítottuk össze, az eltéréseiket (cseréket, törléseket, beillesztéseket) páronként tároltuk, az FR14 kimenetét a 15-ével, valamint a 15-ét a 16-ával összevetve. Bár hasznos lett volna a három OCR-motor eltéréseit összefésülni, hogy együtt lássuk a háromelemű alternatívahalmazokat, ha mindhárom alkalmazás más-más karaktereket lát az oldalon, de ettől ennek bonyolultsága miatt eltekintettünk.

Eltérések összevetése nyelvmodellel Ha egy soron több eltérést is találtunk, akkor ezeket az összes lehetséges, 2^n darab sorváltozattá kombináltuk össze, ahol n az adott soron talált eltérések száma. A hatnál több eltérést tartalmazó sorokat ignoráltuk. Az újságok sorai tipikusan 4–5 szóból állnak, így az egy soron belüli eltérések általában közel esnek egymáshoz. E közelség miatt az eltérések lehetséges kombinációit érdemes vizsgálni, hogy a nyelvmodell helyesen tudjon a változatok között dönteni.

A Pethő és mtsai (2023)-ban ismertetett, szabadon elérhető karakterszintű, kétirányú LSTM alapú, eredeti elektronikus sajtószövegeken tanított nyelvmodellt használtuk, amelyet kifejezetten OCR-szövegek javítására és kapcsolódó feladatokra készítettek a szerzők. Ennek `BiLSTM_Model.predict_subsequences()` metódusát használtuk a sorváltozatok kiértékelésére. A modellel olyan sztringek perplexitását számíttattuk ki, amelyek az adott soron található első eltéréstől az utolsó eltérésig tartanak, amihez hozzávettünk balról 5 és jobbról 5 karaktert. További kontextusként kiegészítettük a vizsgált sorokat a megelőző és a következő sorral (a PyMuPDF által megállapított sorrend szerint), kezelve az elválasztásokat. A legalacsonyabb prediktált perplexitású kombinációt és így az egyes eltéréseknek ezt a kombinációt adó változatát tekintettük helyesnek.

Egy példa:

– **Előző sor:** *A garázsban jó meleg volt. Húsz órákat*
– **Vizsgált sor FR14 ('a' változat):** *dolgozott egyvégtében. A parancsnok*
– **Vizsgált sor FR15 ('b' változat):** *dolgozott egy végiében. A parancsnok*
– **Következő sor:** *megkedvelte. Gyakran benn aludt a finom*

A két eltérés négy lehetséges kombinációját vizsgáltuk. A szögletes zárójelek jelölik azon karakterszekvenciák elejét és végét, amelyekre a nyelvmodellel a perplexitást számítottuk:

aa : `volt. Húsz órákat dolgozott[ egyvégtében.] A parancsnok megkedvelte.`
ab : `volt. Húsz órákat dolgozott[ egyvégiében.] A parancsnok megkedvelte.`
ba : `volt. Húsz órákat dolgozott[ egy végtében.] A parancsnok megkedvelte.`
bb : `volt. Húsz órákat dolgozott[ egy végiében.] A parancsnok megkedvelte.`

A nyelvmodell a szögletes zárójelek közötti karakterek mindegyikének egyenként, az őket körülvevő 30–30 karaktert kontextusként használva prediktálja a feltételes valószínűségét, majd ezek alapján kiszámítjuk a zárójelek közötti részsztring átlagos perplexitását. Itt a modell helyesen az 'aa' változatnak tulajdonítja a legalacsonyabb perplexitást (1,185, szemben az 'ab' 1,216 és a 'ba' 1,383 értékével). Ennek megfelelően az első eltérés 'a' változatát tekintjük helyesnek, tehát a szóköz hiányát a jelenlétével szemben, a második eltérésnek pedig szintén az 'a' változatát, tehát a `t` jobb, mint az `i`.

Míg az előző három feldolgozási szakasz futtatása a korpuszunkon néhány óráig tartott, a sorváltozatok nyelvmodellel történő kiértékelése mintegy három napot vett igénybe egy közepes teljesítményű GPU-s asztali számítógépen.

Validálás A nyelvmodell által az egyes eltérések relatív jóságára adott predikciók helyességét kézzel validáltuk. Három esetet különböztettünk meg:

- A **predikció helyes** ('OK'), tehát valóban a modell által alacsonyabb perplexitásúnak vélt változat a pontos (a fenti példában 'a' és 'a'), míg a másik változat (ott 'b' és 'b') OCR-hiba.
- A **predikció helytelen** ('ROSSZ'), tehát a modell annak a változatnak a perplexitását véli alacsonyabbnak, amely az emberi annotátor megítélése szerint az OCR-hiba.
- A predikció helyessége **eldönthetetlen** vagy **értelmezhetetlen** ('EGYÉB'), mert a nyelvmodellnek átadott mindkét variáns OCR-hiba, vagy nem egyértelmű, hogy milyen karakter áll a szkennelt oldalon az adott helyen.

FR15	FR16	LM szerint	Annotátor	Validálás
`A[lt]aVista`	`A[h]aVista`	15	15	OK
`168 [ó]ra`	`168 [a]ra`	16	15	ROSSZ
`Fés[ő]s`	`Fés[ö]s`	15	–	EGYÉB
`.....[.]`	`.....[]`	15	?	EGYÉB

1. táblázat. A validálás három esete. A harmadik sorban mindkét input OCR-hibás, a negyedik sorban a különbség eldönthetetlen vagy legalábbis irreleváns.

3.3. Összehasonlítás más módszerekkel

Az OCR-hibaelemzés egy klasszikus munkája Rice és mtsai (1999). Szerzői az OCR-fejlesztés hőskorában, 1992 és 1996 között az OCR-programok versenyére

kijelölt, nehéznek szánt autentikus oldalképekre a különböző alkalmazások által generált kimenetet hasonlították össze a helyes (kézzel átírt, ellenőrzött) elektronikus szöveggel, a legalább két alkalmazás által elrontott szövegrészletekből véletlenszerűen kiválasztottak 1000 darabot, végül ezeket kézzel jellegzetes hibatípusokba csoportosították. Itt tehát mind a „gold" szövegváltozat előállítása, mind a hibák kiértékelése számottevő emberi munkát igényelt. Hibaként a helyesnek tekintett változattól való eltéréseket definiálták, ezeket automatikusan, sztringillesztő algoritmussal azonosították.

Laki és mtsai (2022) egy újabb, kifejezetten magyar OCR-javítással foglalkozó munka. A szerzők Mikszáth és Jókai összes műveit dolgozták fel, az OCR-változatot egy (viszonylag) helyesnek tekintett, meglévő elektronikus szöveggel vetették össze. Részletesebben kitérnek azokra a nehézségekre, amelyeket egy OCR-szöveg és a neki sokszor nem is teljesen pontosan megfelelő digitalizált folyószöveg illesztése és összehasonlítása okoz. Emellett megjegyzendő, hogy korlátozott az olyan szövegek száma, amelyek egyszerre rendelkezésre állnak közel hibátlan elektronikus szövegként és OCR-ként. Ez behatárolja a szerzők által javasolt módszert, amelynek lényege, hogy közel hibátlan elektronikus és OCR-szövegekből alkotott párokon tanítanak OCR-hibás szöveget hibátlanná átalakító seq2seq neurális modellt. Az anyagok előkészítése itt is számottevő manuális munkát igényelt, négy annotátor dolgozott 3,5 millió szónyi elektronikus szöveg hibáktól való megtisztításán.

Gupte és mtsai (2021) megoldást javasolnak arra a problémára, hogy viszonylag csekély mennyiségű gold standard elektronikus szöveg és neki megfeleltethető OCR-változat áll rendelkezésre az OCR-javító gépi tanulási modellek tanítására. Szabadon elérhető keretrendszerükkel tetszőleges elektronikus szövegekből tömegesen állíthatók elő olyan oldalképek, amelyeket realisztikusnak szánt filterekkel rontanak el automatikusan, hogy ezáltal szimulálják az OCR alapjául szolgáló szkennelt oldalképek tökéletlenségeit (elmosódott betűk, átszűrődő túloldal, nem tiszta fehér háttér, kifakult, nem kontrasztos betűk stb.).

Megközelítésünk a fentiekkel szemben nem épít gold standard szövegekre, hanem kizárólag oldalképeket és az OCR-alkalmazások által előállított kimenetet használja. Az összehasonlítandó szövegek illesztését a sorok koordinátái alapján hajtja végre, ami többnyire triviális feladat. A sorokon belüli eltéréseket sztringillesztő algoritmus azonosítja, amely a rövid bemenetpárokat gyorsan feldolgozza. Módszerünk egyáltalán nem igényel manuális munkát; a cikkünk készítése során egyedül a modell kimenetét validáltuk kézzel, ami nem szerves része a folyamatnak. Ugyan Gupte és mtsai (2021) OCR-hibák tömeges kezelésére nem használták a keretrendszerüket, erre nyilvánvalóan alkalmazható és teljesen automatizált. Ugyanakkor a mi eljárásunkkal szemben hátránya, hogy kizárólag szintetikusan előállított és manipulált oldalképeken alapul, és ezek formátuma is behatárolt, miközben mi autentikus, sokszor igen összetett szerkezetű, tipográfiai szempontból is változatos anyagot vettünk alapul, ami értelemszerűen pontosabban tükrözi a karakterfeldolgozás során felmerülő hibalehetőségeket.

Végül módszerünk megengedte, hogy az előbbiekhez képest nagyságrendekkel több szöveg feldolgozására alapozzuk következtetéseinket: Rice és mtsai (1999)

270 oldalnyi valós anyagon talált hibák közül 1000-et elemez. Laki és mtsai (2022) a tanítóadatokkal együtt 2 millió sornyi adattal dolgoztak, ezen belül 200 ezer sornyi szövegben kevesebb mint 5000 hibát azonosítottak. Mi kb. 1 millió oldalt, illetve OCR-alkalmazástól függően kb. 200 millió sort vetettünk össze, az ezekben azonosított és automatikusan kiértékelt eltérések száma kb. 15 millió.

Cikkünk nem 2023-ban íródott volna, ha nem említenénk meg ezen a ponton a ChatGPT-t mint lehetséges OCR-javító eszközt. Ha a ChatGPT-t szólítjuk fel a szövegben az OCR-hibák azonosítására és javítására, egyszerűbb esetekben hasonló válaszra számíthatunk, mint az alábbi példában.

Balra a javítandó szöveg, jobbra a ChatGPT által javított párja:

1987. május 28-án alakult meg a Játéktárak Nemzetközi Szövetsége (ITLA) Torontóban. 2011 -ben már négy kontinensen ünnepelték. A kezdeményezés célja a játék értékének megünneplése és a generációk összekötése a játékban. Ezzel egy idó'ben Magyarországon 1987. május 30-án alakult meg a Kiss Áron Magyar Játék Társaság. Célja volt a játékkultúra értékei-nekjelderítése és megó'rzése,Jejlesztése és terjesztése, a játék értékének elismertetése és a gyermekjátékhoz való jogának érvényesítése.

1987. május 28-án alakult meg a Játéktárak Nemzetközi Szövetsége (ITLA) Torontóban. 2011-ben már négy kontinensen ünnepelték. A kezdeményezés célja a játék értékének megünnepelése és a generációk összekötése a játékban. Ezzel egy időben Magyarországon 1987. május 30-án alakult meg a Kiss Áron Magyar Játék Társaság. Célja volt a játékkultúra értékeinek felderítése és megőrzése, fejlesztése és terjesztése, a játék értékének elismertetése és a gyermekjátékhoz való jogának érvényesítése."

Erősen zajos szöveg esetén az emberi olvasó könnyedén felismeri a jól ismert szöveget, míg a ChatGPT teljesítménye nem meggyőző:

Input:	Javított szöveg:
“HazádnahŕendüCetlenűl Légy hive, ó magyar, ßölcsöd az s majdan sírod is, !Mefy ápoC s eltahgr.	"Hazádnak rendületlenül Légy híve, ó magyar, Bölcsőd az s majd a sírod is, Mert apád s ős nagyanyád.
Я nagy világon e hjvűl Nincsen számodra hefy, Áldjon vagy verjen sors kfze, Itt élned, halnodkfH"	Én nagy világon e hűség Nincsen számodra hely, Áldjon vagy verjen sors keze, Itt élned, halnod kell."

A ChatGPT pontosságánál fontosabb szempont a költsége: Az általunk feldolgozott kb. kétmilliárd inputtokennel és kb. ennyi outputtokennel számolva a feldolgozási költség (0,0015$/1000 inputtoken, 0,002$/1000 outputtoken) mintegy 7 millió dollár lenne.

4. Eredmények

4.1. A validálás eredményei

Összesen 2000 eltérésre validáltuk kézzel a nyelvmodell által adott predikciót, az eredmények a 2. táblázatban láthatók.

kategória	db	%	EGYÉB nélkül
OK	1006	50,3%	**89,0%**
ROSSZ	124	6,2%	11,0%
EGYÉB	870	43,5%	–

2. táblázat. A validálás eredménye 1. táblázat kategóriái szerint. A releváns esetek 89%-ában adott jó eredményt a nyelvmodell.

Az alábbiakban a validált anyag néhány során azt mutatjuk meg, hogy a helyes választás milyen eseteivel találkoztunk.

1. Bár az egyik feldolgozott folyóiratban olyan a betűtípus, hogy az a-t jogosan nézi o-nak az egyik alkalmazás, a nyelvmodell felülbírálja ('OK'):
   2005. j**[a/o]**nuár → 2005. j**a**nuár
2. Ritka személynévről sikeresen dönt:
   M**[a/o]**zsaroff Miklós → M**a**zsaroff Miklós
3. Van, hogy arra is rájön, hogy a ritkított szedés a jó megoldás:
   A R Q M A I C I R**[␣/‿]**K U S Z → A R Q M A I C I R‿K U S Z
4. Felismeri, hogy a két írógép-idézőjelnél sokkal jobb megoldás a záró idézőjel és a 11 kombinációja (a számjegy nyilván egy indexszám):
   festői ekvivalenseit is.**[”11/""]** → festői ekvivalenseit is.”**11**

Egy példa tévedésre ('ROSSZ'):
a **[„/.]**Spottdrossel" szóra utal → a **.**Spottdrossel" szóra utal
Egy példa arra, hogy az 'EGYÉB' kategória az esetek többségében nem azt jelenti, hogy az OCR-motor mind a két vizsgált esetben hibázott, csupán annyit, hogy az oldalképtől függően bármelyik változat helyes lehet:
2005.január **[16/3]** → 2005.január **?**

4.2. Irányítatlan eset

Ebben a részben két OCR-rendszer közötti eltéréseket elemzünk. Itt csak azt az információt használjuk fel, hogy mi a két eltérő karaktersorozat, azt nem, hogy melyik lehet a helyes és melyik a helytelen, azaz nem nézzük a tévesztés irányát. Egy részkorpuszon talált leggyakoribb 50 eltérés a 3. táblázatban látható.

A 3. táblázatban azt látjuk, hogy minden esetben van valamilyen alaki hasonlóság a két alak között. Másképp megfogalmazva lényegében mindig egy kicsi nyomdafestékfoltocska ottléte vagy ott nem léte az eltérés. Ez megjelenhet például ékezetkülönbségben, abban, hogy egy betű többre esik szét vagy abban, hogy a papíron véletlenül lévő kis folt írásjelként értelmeződik. Érdekes eset a 34., 37. és 48. elem. Ezek nyilván sor elején/végén vannak, és a felismerésbe belógó függőleges vonalak okozzák. Új eredmény, hogy ezek ennyire gyakori hibák. Kitűnik, hogy a leggyakoribb tévesztések között is szerepel többkarakteres, azaz nyilvánvalóan szükséges ezek kezelése OCR-javítás során.

Bizonyos karakterek nagyon sokféleképpen eltéveszthetők, ezeknek az alakja nagyon sok más karakterre vagy karakterkombinációra hasonlít. Szélsőséges eset az n, leggyakoribb párjai a következők:

1. □ ␣	11. i l	21. □ ’	31. a o	41. c é
2. , .	12. m rn	22. A Á	32. □ i	42. ó’ ő
3. □ .	13. i t	23. □ :	33. b h	43. . :
4. l t	14. ö ő	24. o ó	34. □ I␣	44. ü ű
5. c e	15. " ”	25. ó ő	35. in m	45. m ni
6. □ ,	16. r t	26. E É	36. / l	46. d tl
7. a á	17. i r	27. h li	37. □ |␣	47. m tn
8. - —	18. 1 I	28. ! i	38. ! l	48. □ ␣I
9. i í	19. □ -	29. □ •	39. ii ü	49. 1 l
10. e é	20. I l	30. □ ▶␣	40. ’ ’	50. nn rm

3. táblázat. A leggyakoribb OCR-eltérések. A két karaktersorozatot ábécérendben tüntetjük fel. A □ jel az üres sztringet jelöli.

`n` ↔ `ri rt tt rr u a h r m`

A legtöbb párral bíró többkarakteres elem az `Az`, párjai:

`Az` ↔ `Ki ki Ni fia la fa Nl te fia. Pa`

Bizonyos karaktereknek vagy -sorozatoknak viszont annyira specifikus az alakjuk, hogy lényegében csak egyféleképpen téveszthetők el (4. táblázat).

`rn` `m` 99%	`tn` `m` 95%	`cl` `d` 88%	`nt` `m` 84%
`ó’` `ő` 97%	— `-` 92%	`Á` `A` 88%	`nn` `rm` 82%
`rm` `nn` 97%	`tl` `d` 90%	`in` `m` 84%	`É` `E` 80%

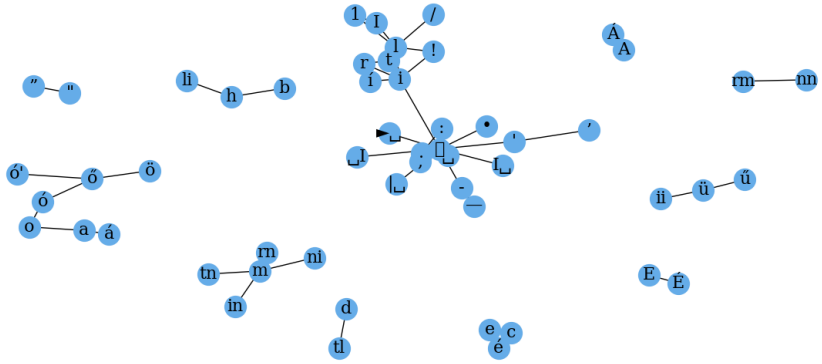
4. táblázat. A legspecifikusabb karaktersorozatok mindig az első oszlopban láthatók. Négy olyat is látunk, ami szinte kizárólag (84–99%-ban) az `m`-mel áll párban, kettőt, ami a `d`-vel. Nagykötőjel szinte csak (92%-ban) kiskötőjelre tud elromlani, és bizonyos nagy ékezetes betűknek is elég egyedi az alakja.

Az egymáshoz kapcsolódó elemeket gráfként ábrázolva (1. ábra) szépen megjelennek a hasonló alakú karakterek és karaktersorozatok részgráfjai: balra a kör-, középtájon a pont-, fölötte a függőleges vonal alakú karakterek foglalnak helyet.

4.3. Irányított eset

A 4.1 részben láttuk, hogy az alkalmazott nyelvmodell megbízhatóan meg tudja mondani, hogy egy OCR-eltérés két tagja közül melyik a helyes. Erre építve megtehetjük, hogy egyszerűen a nyelvmodell által adott perplexitás alapján fogadjuk el az egyik tagot helyesnek, a másikat pedig helytelennek. Így a 4.2. részben leírt „egyiket a másikra rontja” (`{m,rn}`) halmazok helyett „a-t b-re rontja” (`m`→`rn`) típusú irányított információnk lesz, ezt az irányított adathalmazt vizsgálhatjuk. A 2. ábra mutatja be az ennek megfelelő irányított hálózatot.

A 2. ábrát megvizsgálva látjuk, hogy a leggyakoribb hiba a szóköz elmaradása. Teljesen szimmetrikus a pont és a vessző kapcsolata. Aszimmetriára példa



1. ábra: Az egymás OCR-tévesztéseiként előforduló elemek hálózata egy részkorpuszon. Az adatok a 3. táblázatból valók. Minél rövidebbek az élek, annál gyakoribb az adott kapcsolat. Középen a □ jel az üres stringet jelöli.

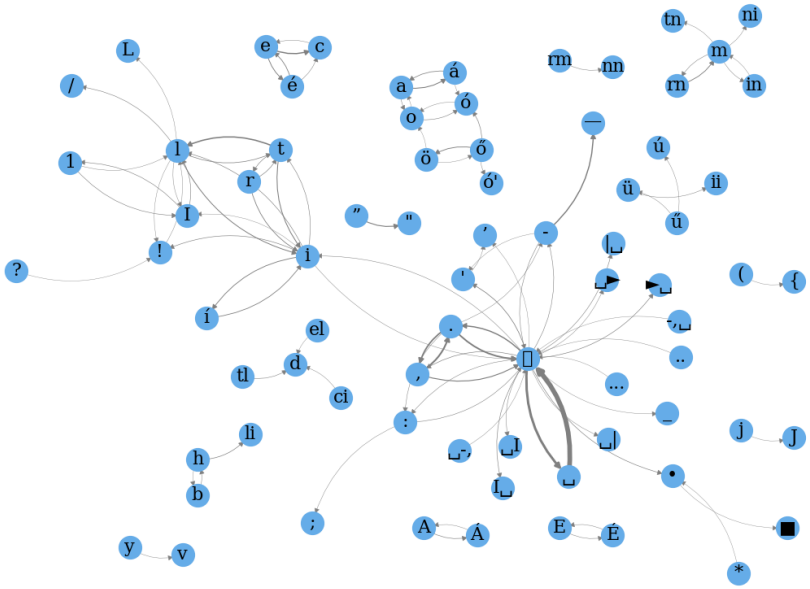
a kiskötőjel és a nagykötőjel viszonya: a nagykötőjel alakja annyira eltér a többi karaktertől, hogy nagyon ritkán romlik el, a kiskötőjelből viszont elég gyakran nagykötőjel lesz. Szintén aszimmetrikus az `y`→`v` viszony, az `ü`→`ii` kapcsolat az `ii` ritkasága miatt, valamint a két idézőjel viszonya. Míg az `e` és az `é` kapcsolata szimmetrikus, a `c`-vel való kapcsolatuk már eltérő. Létraszerű struktúrát ad az `a-o-ö` és az ékezet viszonya. A kétkarakteres eltérések nagy részében a kettéválás a jellemző (`m`→`tn`, `m`→`ni`, `m`→`in`, `h`→`li`) de előfordul összeolvadás is (`el`→`d`, `tl`→`d`, `ci`→`d`).

A módszerünk implementációját és a teljes hibalistákat az alábbi repozitóriumban tesszük közzé: https://github.com/ril-lexknowrep/ocr-error-stat.

4.4. OCR-alkalmazások hibázási gyakorisága

A nyelvmodell általi predikciók azt is megengedik, hogy nagy mennyiségű változatos és autentikus adaton összevessük a három alkalmazás hibázási gyakoriságát. Ehhez egyszerűen megszámoljuk, hogy az összes eltérés közül hány esetben prediktálja jobbnak a nyelvmodell az egyik alkalmazás outputját, mint a másikét. Ha az FR14 és az FR15 kimenetét hasonlítjuk össze, az FR14 változata 3,7 millió, az utóbbié 3,9 millió esetben nyert. Az FR15 és FR16 közül az FR15 6,35 millió, az FR16 6,2 millió esetben jobb. E nagyon nagy eltérésszámnak bizonyosan nem elhanyagolható része nem OCR-, hanem illesztési hiba, és a validálás szerint kb. a fele olyan eset, ahol két hibás vagy eldönthetetlen változatot hasonlíttattunk össze a modellel. Mindazonáltal megállapítható, hogy a FR15 outputja valamivel megbízhatóbb mind a FR14-énél, mind a FR16-énál.

A három alkalmazás pontosságát úgy is ellenőriztük, hogy néhány újságszám PDF-éből kinyert folyamatos szöveg teljes perplexitását vetettük össze illesztés nélkül. Itt szintén a FR15 outputja adta a legkedvezőbb, a FR14 a legrosszabb eredményt. Azonos sorrendet kapunk, ha az utóbbi szövegeket az emMorph-fal



2. ábra: A 100 leggyakoribb OCR-tévesztés irányított hálózata egy részkorpuszon. Minden él egy helyes elemből egy hibásba mutat, amire el szokott romlani. Minél vastagabbak az élek, annál gyakoribb az adott kapcsolat. Sok esetben oda-vissza is van nyíl két elem között. Középen a □ jel az üres stringet jelöli.

(Novák és mtsai, 2016) feldolgozva az elemezhetetlennek ítélt szótokenek száma alapján próbáljuk alulról becsülni az OCR-hibák számát.

Köszönetnyilvánítás A modellek tanítását az HUN-REN Cloud infrastruktúrán futtattuk. Köszönetet mondunk az HUN-REN Cloud (lásd: Héder és mtsai, 2022; https://science-cloud.hu/) használatáért, ami hozzájárult a publikált eredmények eléréséhez.

5. Összefoglalás

Cikkünkben bemutattuk, hogy milyen OCR-hibák detektálhatóak úgy, hogy összehasonlítjuk azonos oldalképekre több OCR-alkalmazás által készített outputot. Gyakori, hogy egy adott karaktert vagy karaktersorozatot az egyik OCR-motor tévesen azonosít be, egy másik viszont helyesen. Nyelvmodell segítségével nagy biztonsággal meg tudjuk állapítani, hogy a kettő közül melyik a helyes. Az eljárást nagyon nagy mennyiségű adaton futtatva gazdag adathalmazt gyűjtöttünk az OCR-hibákról, amelyből általános jellegzetességeket állapítottunk meg.

Ha lényegében ugyanezt a módszert nem a hibák detektálására, hanem javítására használnánk, ez egyértelműen a zaj nagymértékű csökkenését, a digitalizált

dokumentumok tartalmának az eredeti változathoz való közelítését eredményezné: Ha a különböző OCR-motorokkal előállított alternatívák között már van egy helyes, akkor a használt nyelvmodell közel 90 százalékban azt választja. Emellett a hibastatisztikák ismeretében újabb nyelvmodellek célzottan taníthatók a gyakori OCR-hibák kiszűrésére, valamint egy a nyelvmodellt kiegészítő hibajavító algoritmus automatikusan javasolhatja a helyes alternatívát olyan pontokon is, ahol az OCR-motorok mindegyike (bár másképp) téved.

Hivatkozások

Bassil, Y., Alwani, M.: OCR context-sensitive error correction based on google web 1t 5-gram data set. American Journal of Scientific Research (2012)

D'hondt, E., Grouin, C., Grau, B.: Generating a training corpus for OCR post-correction using encoder-decoder model. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers). p. 1006–1014. Asian Federation of Natural Language Processing (2017)

Gupte, A., és mtsai: Lights, camera, action! A framework to improve NLP accuracy over OCR documents. CoRR abs/2108.02899 (2021), https://arxiv.org/abs/2108.02899

Héder, M., Rigó, E., Medgyesi, D., Lovas, R., Tenczer, S., Török, F., Farkas, A., Emődi, M., Kadlecsik, J., Mező, G., Pintér, A., Kacsuk, P.: The past, present and future of the ELKH Cloud. Információs Társadalom 22(2), 128–137 (2022)

Kamlah, J., Stegmüller, J.: Ocromore - Combining multiple OCR-engine results to improve character recognition accuracy (Nov 2018), https://doi.org/10.5281/zenodo.1493860

Kissos, I., Dershowitz, N.: OCR error correction using character correction and featurebased word classification. In: 2016 12th IAPR Workshop on Document Analysis Systems (DAS). pp. 198–203 (2016)

Laki, L.J., és mtsai: OCR-hibák javítása neurális technológiák segítségével. In: XVIII. Magyar Számítógépes Nyelvészeti Konferencia. pp. 417–430. Szegedi Tudományegyetem, Szeged (2022)

Novák, A., Siklósi, B., Oravecz, C.: A new integrated open-source morphological analyzer for Hungarian. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), Paris, France (2016)

Pethő, G., és mtsai: Korpusztisztítás és sorvégi kötőjelek kezelése karakteralapú neurális nyelvmodellel. In: XIX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 291–304. Szegedi Tudományegyetem, Szeged (2023)

Rice, S.V., Nagy, G., Nartker, T.A.: Optical character recognition: an illustrated guide to the frontier. Kluwer, New York (1999)