6.3. Az algoritmus

A helyreállítást megvalósító algoritmus a következő lépésekből áll.

1. Előkészítjük az adatmezőhöz rendelt eszközkészletet.
2. Átírjuk az adatot a laza átíróval.
3. Az így kapott reguláris kifejezést illesztjük a listára.
4. Megtalálható a listán így? Ha igen, visszaadjuk az összes találatot. ✓
5. Ha nincs, akkor közelítő kereséssel keressük a szigorú átiratot a listán.
6. Megtalálható? Ha igen, visszaadjuk a legjobb találatot. ✓
7. Egyébként: visszaadjuk a szigorú átiratot. ✓

Az algoritmusnak három (pipával jelölt) kimeneti pontja van, ezek rendre megfelelnek az #**1**+#**2** szintnek, a #**3** szintnek illetve annak az esetnek, amikor semmilyen módon nem sikerült a megfelelő lista egy elemeként azonosítani a helyreállított alakot, így a puszta szigorú átiratnál jobbat nem tudunk mondani. 'Имре' → Imre esetén már az #**1** szint eredményt adna. Ha 'Андрош' a kiinduló adat, akkor reguláris kifejezés segítségével találjuk meg a helyes András alakot (vö: 2. táblázat). 'Форенц' bemenetnél a regex nem segít, mert az orosz 'о'-nak a laza átíró nem felelteti meg az e-t. Itt a közelítő keresés találja meg a Ferenc alakot. Végül, ha a 'Момольсильтер' szóalak az adat, akkor egyik megközelítés sem jár sikerrel, így csupán a szigorú átíróval létrehozott Momolszilter alakot tudjuk visszaadni.

A módszert python nyelven implementáltuk, a 5. lépésben a közelítő keresést a `difflib` csomaggal valósítottuk meg. Az 4. lépésben a kapott találati halmaz sorrendezéséhez két szempontot veszünk figyelembe: egyrészt a találatoknak a szigorú átirathoz való hasonlóságát, valamint lehetőség szerint a találatok általános gyakoriságát. Előbbit a `difflib` megfelelő függvényével számítjuk ki, utóbbi adat a vezetéknevek és a keresztnevek esetében állt a rendelkezésünkre egy első világháborús veszteséglista formájában. A találatokat a két adatból képzett közös pontszám sorrendjében, a pontszámmal együtt adjuk vissza. Az 'Андрош' alakból laza átíróval képzett regex – `(A|Á)(n|m)(d|gy|t)(r|l)(a|á|o|e)(s|sch)` – 192 különböző alakot fed le. Erre a lazaságra általában szükség is van, mert az egyes adatelemek nagyon sok változatban valóban előfordulnak.

6.4. Idegennyelvű szövegek

Az eredeti kiindulópontunkkal szemben, miszerint *magyar* nyelvű adatokat kell helyreállítanunk, kitűnt, hogy egyéb célnyelv (vö: 3. rész) is előfordul. Például a német. Egyrészt számos német nemzetiségű hadifogoly is volt, másrészt sokakat akkor fogtak el, mikor a front már Ausztria területén járt, így az elfogás helye osztrák település (6. táblázat).

A német adatelemek kezelésére szükséges volt létrehozni egy komplett *orosz–német* átíró szabályrendszert, amiben a németnek megfelelő szabályok kaptak helyet: 'ц' → c helyett 'ц' → z; 'в' → v helyett 'в' → w; 'ай' → aj helyett 'ай' → ei; 'ой' → oj helyett 'ой' → eu stb.

Гроц	→ Graz
Лииц	→ Linz
Фрайштадт	→ Freistadt
Дойчландберг	→ Deutschlandsberg
Штокаров	→ Stockerau
Цветел	→ Zwettl

6. táblázat. Osztrák településnevek az adatbázisban.

A rendszer működését alkalmassá tettük arra, hogy bizonyos feltétel teljesülése esetén alternatív eszközkészlet használatára lehessen váltani. Azon helyek esetén tehát, melyekben szerepel az 'Австрия' alak, átváltunk az orosz-német szabályok + osztrák településlista eszközkészletre.

7. Eredmények, megfontolások

7.1. Kiértékelés

Az előzőekben bemutatott eljárás kiértékelésére két mértéket használtunk. A *megalapozott helyreállítások aránya (M)* egy fedés jellegű mérték, azt mutatja meg, hogy az adatok hány százalékára tud a módszerünk a buta szigorú átírásnál jobb megoldást adni (ld. az algoritmus 4. és 6. pontja a 6.3. részben). A *helyes helyreállítások aránya (H)* egy pontosság jellegű mérték, azt mutatja meg, hogy az adatok hány százalékára van valóban helyes helyreállítás. A nagyon specifikus feladat miatt más módszerekkel való összevetésre nincs lehetőség. Az M és H értékeket a 7. táblázat mutatja be. Az adatok kezelhetőség szempontjából az M érték alapján kétfelé oszlanak: a nevek és az országok (✓a táblázatban) jól kezelhetők (M = 95-100%), a többi helyadat sokkal nehezebb (M = 50-70%). Utóbbiban valószínűleg közrejátszanak a felbontás nehézségei (vö: 6.1. rész).

A H értékeket is figyelembe véve négy csoport látszik. A keresztnevek és az országnevek (2-4. és 8. sor) szinte mindegyikére van ajánlata az algoritmusnak és lényegében az összes ajánlat helyes is. A vezetékneveknél (1. sor) a helyesség alacsonyabb. Közrejátszik, hogy keresztnevekhez képest sokkal (∼50x) több vezetéknév van, jóval több nehezen megfejthető példa fordul elő: 'Хумато' → Homoga?, 'Турuел' → Turul?; valamint, hogy a *B. Kovács* típusú összetett vezetéknevek nincsenek jelenleg kezelve. A megyéknél (5. sor) kevesebb helyen tud tippet adni az algoritmus, olyankor viszont szinte mindig helyes a tipp. Itt jellemző hiba, hogy keverednek a különböző méretű közigazgatási egységek: *Derecske* vagy *Gyömrő* például megjelenik megyeként is az adatbázisban. A település és a járás (6-7. és 10-11. sor) a legnehezebb. A járás kisebb fontosságú és eléggé ritkán is fordul elő, a település viszont kiemelten fontos a hadifoglyok azonosításhoz. Itt sajnos az M és a H is alacsony, leginkább ezen szükséges javítani a jövőben. A nehezen megfejthető példák (pl.: Фоло, Улалануш) mellett gondot jelentenek itt a nem-osztrák külföldi települések, valamint a gyakran előforduló hosszú te-

	adatmező	M	helyes	/ összes	= H	H/M
✓ 1.	vezetéknév	95,8%	76	100	76%	79%
✓ 2.	keresztnév	95,0%	92	100	92%	97%
✓ 3.	apai keresztnév	95,2%	70	78	90%	95%
✓ 4.	születés: ország	99,9%	45	45	100%	100%
5.	születés: megye	70,7%	32	49	65%	92%
6.	születés: járás	60,9%	7	15	47%	77%
7.	születés: település	65,9%	31	61	51%	77%
✓ 8.	fogságba esés: ország	99,9%	33	33	100%	100%
9.	fogságba esés: megye	67,7%	11	12	92%	—
10.	fogságba esés: járás	46,2%	1	4	25%	54%
11.	fogságba esés: település	67,5%	29	56	52%	77%
	összesen	**85,5%**	**427**	**553**	**77%**	**90%**

7. táblázat. Eredmény: az adatok 77%-ához tudunk helyes helyreállított alakot rendelni. A kiértékelésben a megalapozott helyreállítások aránya (M, „fedés”) és a helyes helyreállítások aránya (H, „pontosság”) szerepel. Előbbit a teljes, 682000 rekordot tartalmazó adatbázis alapján számoltuk, utóbbit az adatbázisból képzett 100 rekordból álló random mintán állapítottuk meg manuális kiértékeléssel. Az *összes* mező mutatja, hogy 100 sorból hányban volt jelen a szóban forgó adat. A H/M arány arról informál, hogy a megalapozott helyreállítások hány százaléka helyes valóban. A 9. sorban – vélhetően a H-hoz használt minta kis mérete miatt – nem értelmezhető érték adódik.

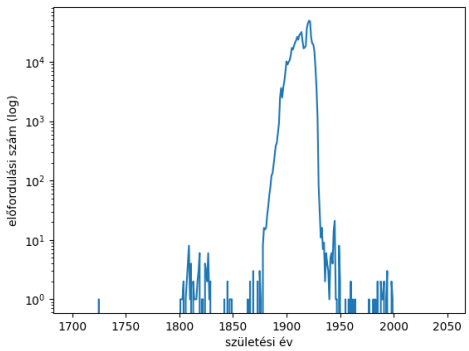
lepülésnevek, melyek gyakran számos hibát tartalmaznak (pl.: ’Яскорогенуй’ → Jászkarajenő?, ’Пишпекляний’ → Püspökladány?).

Látjuk, hogy az M és H értékek sok helyen összecsengenek: ahol tud valamit mondani az algoritmus, ott legtöbb esetben helyes is a javasolt helyreállítás. Az M-kiértékelés automatikus (össze kell számolni), a H-kiértékelés manuális munkát igényel. A H/M arány megmutatja, hogy az algoritmus által szolgáltatott megoldások mennyire jók. Ha ez magas, annak az az előnye, hogy megspórolhatjuk a munkaigényes H-kiértékelést, mert ekkor a H érték jól becsülhető az M értékkel. Ez az eset azokra az adatmezőkre jellemző, ahol lehetséges helyes adatok száma alacsony.

7.2. A szabadszöveges adatbázisokról

A kiinduló adatbázisunk egy *kézzel készült, korlátozatlan, szabadszöveges* adatbázis. Ez azt jelenti, hogy az adatbevivő nincs semmilyen értelemben korlátozva – például legördülő menüből való választás vagy típusellenőrzés révén –, azaz teljesen szabadon azt ír be az adatmezőkbe, amit csak akar. Az ilyen adatbázisok szükségszerűen következetlenek, mivel nincs olyan mechanizmus, ami biztosítaná az adatok egységességét: hogy ugyanazt mindig ugyanúgy jelöljük, az eltérő dolgokat pedig mindig eltérően.

Amellett, hogy az ilyen adatbázisokban egy adat több validnak mondható formában fordul elő, az ilyen adatbázisokba ellenőrzés híján számos hiba, elírás is belekerül. Azt látjuk, hogy ha nem legördülő menüből kell választani, akkor még a születési év adatot is el lehet rontani (2. ábra). A tanulság az, hogy az adatbázisok készítésekor szükséges az ellenőrzés, az egységesítő mechanizmus.



2. ábra: A adatbázisban szereplő *születési év* adatmező értékeinek eloszlása. A második világháborús hadifoglyok adatai között előfordul 1725-ös és 1999-es születési év is.

Viszont van olyan eset is, amikor valóban szabad kezet akarunk adni az adatszolgáltatónak/adatrögzítőnek. Véleményünk szerint ilyen eset a közvéleménykutatás. Azt gondoljuk, hogy ha egy közvéleménykutatási kérdés esetén – főként ha *miért*-es kérdésről van szó – a válaszadónak néhány előre megadott választási lehetőség közül kell választania, akkor a kutatás szükségszerűen veszít a hitelességéből ahhoz képest, ha a válaszokat szabadon fogalmazhatja meg, például mivel adott esetben véleményét jól visszaadó válasz egyszerűen nem szerepel a lehetőségek között.

A korlátozott módon készülő adatbázisokat persze sokkal könnyebb kiértékelni. Ugyanakkor a korlátozatlan adatbázisok adatainak értelmezése is megvalósítható: nyelvtechnológiai eszközökkel. Két esetben kaphat tehát szerepet a nyelvtechnológia: amikor nem történt előzetes adatellenőrzés/korlátozás (pl. a jelen tanulmányban tárgyalt hadifogoly-adatbázis) illetve amikor nem akarunk előzetes adatellenőrzést/korlátozást (pl. közvéleménykutatás).

A szabadszöveges adatbázisok értelmezési-feldolgozási munkálatait érdemes három szakaszra bontani: (1) adatvizsgálaton alapuló szakasz; (2) gyakorisági hibaelemzésen alapuló szakasz; (3) manuális szakasz. Az első szakaszban valamilyen automatikus rendszer áll elő, ami az adatok jelentős részét kezelni képes, a második szakaszban ezt finomítjuk a felfedett gyakori hibák javítása révén. Tudva azt, hogy ha a tökéleteshez közeli eredményt szeretnénk, akkor nem le-

het megspórolni a manuális szakaszt, a második szakaszban azokkal a hibákkal foglalkozunk, amelyek javítása a legnagyobb haszonnal jár.

A jelen tanulmányban feldolgozott adatbázis a többrétű torzulás miatt a szabadszöveges adatbázisoknak is a szélsőségesen következetlen és sokféle hibával teli fajtájához tartozik. Kezeléséhez a fent (6. rész) ismertetett szabályalapú megközelítést alkalmaztuk. Azért fogtunk hozzá így, mert egyrészt egy teljesen egyedi feladat konkrét problémáit kellett megoldani behatárolt méretű adathalmazon, valamint tanulóadat híján a gépi tanulási módszerek alkalmazására nem volt lehetőség. Ilyenkor ma is lehet létjogosultsága a szabályalapú módszereknek.

7.3. Példák

A 8. táblázatban egy engedéllyel közzétett valódi teljes példa látható.

vezetéknév	**Галь**	**Gál**
keresztnév	**Тибор**	**Tibor**
apai keresztnév	**Эмиль**	**Emil**
születési év	1915	1915
születés helye	**г. Сольнок**	**Szolnok település**,
	ул. Санопи, 17	Szanopi (?) utca 17.
fogságba esés helye	**г. Цветел, Австрия**	**Ausztria, Zwetel település**
nemzetiség	венгр	magyar
fogságba esés ideje	12.05.1945	12.05.1945
elbocsátás ideje	08.07.1947	08.07.1947
fogadó tábor	сдан лагерь № 36	36-os tábor
rendfokozat	лейтенант	hadnagy
elbocsátó tábor	лагерь № 313	313-as tábor

8. táblázat. A helyreállító rendszer által kezelt adatok vastagítva láthatók. Egy helyen nem tökéletes a megoldás: a *Zwetel* helyesen *Zwettl* lenne.

Annak illusztrálására, hogy valóban előfordulhattak félreolvasási hibák (vö: 4. oldal) a kartonok elektronikus rögzítésekor, bemutatunk egy eredeti kartont (3. ábra).

A végső manuális szakaszra maradó adatok helyreállításának nehézségét két példán mutatjuk be. Ilyenkor előfordul, hogy egy-egy adat megfejtése önmagában kutatómunkát illetve több kutató együttműködését igényli. A 4. táblázat 8. bejegyzéseként látható Блодентмигайн helyreállítva Bűdszentmihály. Itt kezelni kell a szó végén lévő ’йн’ variációt, a kieső sz-t, valamint meg kell fejteni, hogy hogyan változhatott az ű az orosz ’ло’ betűkapcsolattá. Ez az eset mindkét torzulástípust példázza, ugyanis minden valószínűség szerint az adatrögzítéskor lett ű-ből ’ю’ megfelelőbb orosz betű híján; majd a digitalizáláskor ’ю’-ból ’ло’ félreolvasás révén. A 6.3. részben idézett Момольсильтер helyreállítva Mosonszentpéter. Itt az segített, hogy az adott napon Mosonszentpéteren esett fogságba


КАРТОЧКА ИНТЕРНИРОВАННОГО
Форма № 2
Пол Батальон № Рота №
1. Фамилия
2. Имя
3. Отчество
4. Год рождения
5. Место рождения
6. Последнее место жительства
7. Национальность
8. Вероисповедание
9. Партийность
10. Подданство (гражданство)
11. Профессия и специальность
12. Образование: а) Общее
б) Специальное
в) Военное
13. Дата интернирования « » 194 года

3. ábra: Egy eredeti karton. Az írás értelmezése nagy gyakorlatot igényel.

a hadifoglyok nagy része, és az adatbázisban szerepeltek az idézett orosz alakra sokban hasonlító, de könnyebben megfejthető verziók is.

Köszönjük Nyéki Bence, Orosz Ferenc, Beke Gábor és Szatucsek Zoltán közreműködését a munkálatokban, illetve hozzájárulásukat a fenti példák megoldásához.

8. Elérhetőség

A helyreállító rendszer részletes technikai információkkal, teljes szabályrendszerekkel, listákkal, futtatható programmal és minden egyébbel elérhető a `https://github.com/dlt-rilmta/hadifogoly-adatbazis` címen. Az eredeti adatokat ez a repozitórium nem tartalmazza, rendelkezésre áll viszont egy adatmezőnként külön-külön randomizált adatfájl (`data/pseudo_1000_42.csv`), amin az eljárás a `make transcribe FILE=pseudo_1000_42` paranccsal futtatható.

9. Továbbfejlesztési lehetőségek

Az ismertetett módszer nem oldja meg maradéktalanul a kitűzött feladatot. Konkrét feladat lévén az elvi cél a teljes, 100%-os megoldás, ehhez, ahogy említettük, mindenképpen szükséges egy manuális munkaszakasz is. Ugyanakkor a teljesítmény növelése érdekében számos továbbfejlesztési lehetőség adódik.

A településlisták jelenleg magyar és osztrák településeket tartalmaznak. Sokszor előfordulnak hiányzó települések. A „magyar verziókat" (Bécs, Bukarest stb.) a magyar (átírószabályokhoz tartozó) listához szükséges hozzátenni, az idegennyelvűeket külön listákba szükséges gyűjteni, és a 6.1. rész elején említettek miatt nyelvenként kezelni. Ehhez minden nyelvhez új átíró szabályrendszer szükséges, valamint egy módszer az aktuális nyelv meghatározására.

Megfontolható, hogy hogyan lehetne automatizálni a fogságba esés idejére alapuló a 7.3. rész legvégén említett ötletet a településnevek jobb azonosítására.

Érdemes lehet újból felmérni a hagyományos vagy neurális gépi tanulás alkalmazhatóságát. Manuális úton természetesen lehet tanítóadatot gyártani, ezzel eddig nem foglalkoztunk. A kiinduló adat ismertetett extrém tulajdonságai miatt azonban nem vagyunk meggyőződve arról, hogy a gépi megközelítés esetünkben jelentősen jobb eredményt adna.

10. Összefoglalás

Tanulmányunkban egy orosz-magyar átíró és helyreállító rendszert mutattunk be, melynek célja a magyar hadifoglyok adatbázisából kiindulva értelmezni és helyreállítani az eredeti magyar adatokat a cirill betűs forma alapján. A kidolgozott szabályalapú módszer az adatok 77%-ához tud helyes helyreállított alakot szolgáltatni. Ez a kiinduló adatbázis minőségét tekintve megfelelő eredmény. A teljesítmény növelésére elsősorban a helyadatok feldolgozásában nyílik lehetőség a jövőben.

Az adatbázis jól példázza a kézzel készült, korlátozatlan, szabadszöveges adatbázisok szükségszerű következetlenségét; az automatikus feldolgozást esetünkben még az adatok kétszeres torzulása is nehezíti. Az ilyen adatbázisok feldolgozását éppen a nyelvtechnológiai eszközök használata tudja megfelelően támogatni.

Hivatkozások

Bradley, J.: The Mari web project's orthography helper(s) (2020), https://www.univie.ac.at/maridict/site-2014/transcription-general.php

Haader, L.: Az ómagyar kódexek hibatipológiájának kutatásáról. In: Korompay, K., Stemler, Á., Terbe, E., C. Vladár, Z. (szerk.) Forráskutatás, forráskiadás, tudománytörténet II., pp. 23–33. Magyar Nyelvtudományi Társaság (2014)

Katona, P., Szikla, G.: A Magyar Nemzeti Levéltár Hajdú-Bihar Megyei Levéltár adatbázisai. Új Nézőpont 1, 61–81 (2014)

Morse, S.P.: Searching the Gulag database in one step (2005), https://stevemorse.org/russian/gulag.html

Prószéky, G., Naszódi, M., Kis, B.: Recognition assistance - treating errors in texts acquired from various recognition processes. In: COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes. pp. 1263–1267. Taipei, Tajvan (2002)

Zoltán, A.: A cirillbetűs írású szláv nyelvek szavainak és neveinek magyar helyesírása. MTA I. Osztály Közleményei 32, 171–192 (1981)

emPhon: Morphologically sensitive open-source phonetic transcriber

Kulcsár Virág[1⋆], Lévai Dániel[12⋆]

[1]HAS-BME Lendület Language Acquisition Research Group
[2]Alfréd Rényi Institute of Mathematics
{kulvirag@gmail.com, levaid@renyi.hu}

Abstract. We propose a new emtsv module which can provide phonetic transcription based on the emMorph state-of-the-art morphological analyzer. In the first part of the paper, we present the motivation, the main problem and the method we are using to leverage Hungarian phonetic transcription with the use of morphological analyses. The second part is about evaluation both intrinsically and extrinsically – we evaluate our transcriber based on the IPA transcriptions of Wiktionary and as part of a speech synthesizer system. The code and models are fully open-source and are available under LGPL 3.0 license at `https://github.com/levaid/emPhon`.
Keywords: phonetics, IPA, emtsv, morphology, analysis, speech, synthesis

1 Introduction, motivation

Hungarian is a highly phonetic language in its orthography. Compared to English, where the lack of productive morphology and the highly irregular orthography makes it hard for any rule-based transcriber to work efficiently, Hungarian orthography is much more informative. For a great extent, simple words like ‘alma‘ *apple* are easy to phonetically transcribe: we can look up the IPA for Hungarian and just substitute the letters one-by-one, giving /ɒlmɒ/ as pronunciation. In most cases, however, it is needed to take into account the Hungarian phonology - there are large set of rules which are governed by not only the interaction of different phones, but by the morphology as well.

The simplest problem that a phonetic transcriber has to overcome in Hungarian is the handling of digraphs (e.g. *cs*, *ny*) and their long counterparts. This is not a trivial task in itself, even though it looks like an innocent problem. One could argue that a modern subword tokenizer is easily able to distinguish between digraphs and and co-occurrences, but that is simply not true. Consider the following example, where the *sz* has been segmented into *-s_ z-*: *als_ zo_ tok* ‘sleep-2PL’, produced both by HuBERT’s tokenizer (Nemeskey, 2020) and by XLM-Roberta’s (Conneau et al., 2020) tokenizer. This example demonstrates

⋆ equal contribution