Szöveg- és Webbányászat házi feladat beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Szántó Tamás, Benda Krisztián**
Neptun-kód:	**ET7D8H, J1CEI3**
Ágazat:	**Adat- és Médiainformatika**
E-mail cím:	**tmas.szanto@gmail.com, krisztianbenda@gmail.com**
Konzulens(ek):	**Dr. Szűcs Gábor**
E-mail címe(ik):	**szucs@tmit.bme.hu**

Téma címe: Névelem felismerés

Feladat

A névelem-felismerést (named entity recognition) segítségével kinyerhetők egy adott korpuszon belül előforduló névelemek, s ezen belül a tulajdonnevek (személynevek, helyek, szervezetek és egyéb tulajdonnevek). A feladat angol nyelvű szövegben 7 típusú entitásnak a felismerése, melyek a következők:

event = esemény; geo = földrajzi entitás; gpe = geopolitikai entitás; obj = objektum, műtárgy; org = szervezet; per = személy; time = idő

A felismerendő entitások állhatnak 1 vagy akár több szóból is. Minden esetben az entitás első szavát külön detektálni kell, ennek jelzésére a B betű használandó (beginning): így B-event, B-geo, stb. címkékkel kell a megfelelő szavakat ellátni. Ha az entitások több szóból áll, akkor az összes többi I-vel jelölendő (inside), azaz I-event, I-geo, stb. címkék; így összesen az egyéb (O) címkével együtt 15 osztálycímke adódik.

Példa:

Indian border security forces are accusing their Pakistani counterparts of lobbing at least four rockets into northern Punjab state.

Indian: B-gpe, Pakistani: B-gpe, Punjab: B-geo, a többi pedig O címkéjű.

2018/2019. 1. félév

1. Megoldási terv

Félév elején a házi feladat megoldási tervünket a következőképpen foglaltuk össze:

Vállalt részfeladatok:

1. Létező megoldások vizsgálata és kipróbálása
2. Órán tanult módszerek áttanulmányozása
3. A tapasztalatok alapján prototípus elkészítése
4. Az elkészült megoldás javítása, továbbfejlesztése
5. Bemutató elkészítése és előadása

Megoldási ötletek:

- Az általánosabb feldolgozási folyamat a következő:
  - Tokenizálás, normalizálás/szótövezés, névelem detektálás, névelem normalizálás
- Névelem detektálására az alábbi megközelítéseket ismerjük
  - Szótár alapú
    - Összes entitás összes formáját össze kell gyűjteni
    - Nagy tudásbázis vagy annotált korpusz szükséges
  - Szabály alapú
    - Mintákat kell írni az entitások illesztéséhez
    - Téma specifikus tudás szükséges
  - Statisztikai modell alapú
    - Valószínűségek hozzárendelése a szövegrészekhez
    - Sok tanuló példány szükséges
    - Előny: téma független tudás
- Ezen detekciók használatának előnyeit fogjuk felmérni és ezalapján a legmegfelelőbbet kiválasztani.

Létező eszközök, módszerek:

- spaCy: python library, statisztikai modell alapú NER, sokféle kategória támogatott, továbbtanítható saját kategóriákkal
- Stanford NER is a Named Entity Recognizer: Java library, kevés alapból támogatott kategória
- Named-Entity-Recognition-BLSTM-CNN-CoNLL: implementáció ehhez a cikkhez, Keras

Használni tervezett technológiák:

- Elsősorban Python 3-at szeretnénk használni
- Kisebb részfeladatok/algoritmusok kipróbálásához opcionálisan RapidMinder-t is igénybe vennénk
- A párhuzamos munkavégzést a GitHub segítségével oldanánk meg

2. Névelem felismerés házi feladat dokumentáció

A féléves munkánkat az alábbi négy részre lehet osztani:

- Feladat megértése, adathalmaz tanulmányozása
- Névelem felismerő rendszer kiválasztása
- SpaCy tanításának és használatának kipróbálása első megoldás elkészítése
- Másodikkörös tanítás és az eredmény javítása

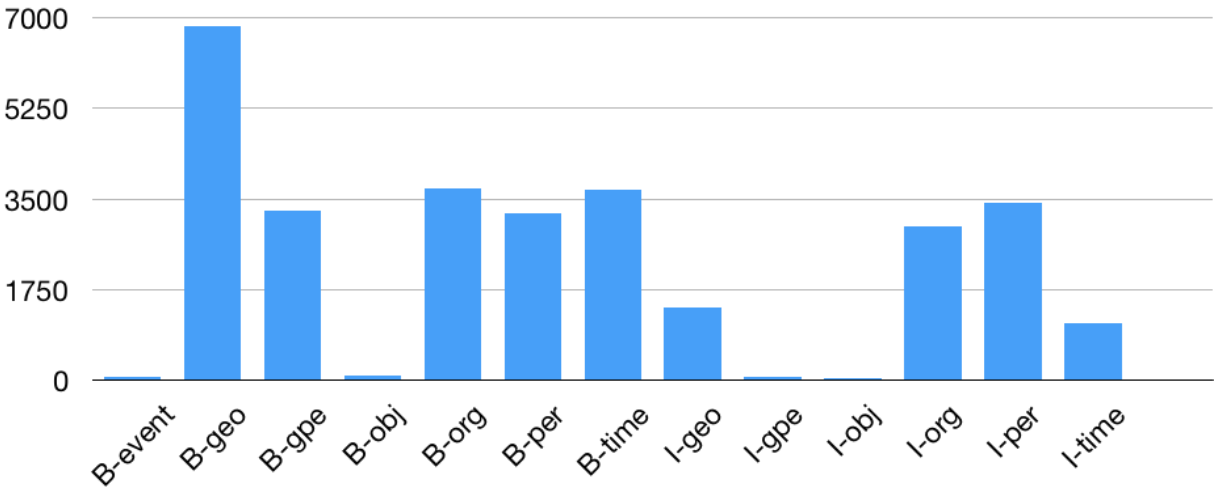
A beszámolónkat is ebben a sorrendben készítettük el, kiegészítve egy összefoglalóval.

2.1. Feladat megértése, adathalmaz tanulmányozása

A hivatalos házi feladat leírását már ismertettük a címlap Feladat résznél, azonban egy adatokkal manipuláló problémához szorosan hozzátartozik az adathalmazok értelmezése és felépítése. Elsőként az elérhető adatokat mutatnánk be.

A házi feladathoz három adathalmaz állt rendelkezésre: TrainNER, Test1NER és Test2NER. A TrainNER négy, a Test1NER három oszlopból áll, míg a Test2NER kettőből. Mindhárom esetben kerek mondatokból álló szöveget tartalmaznak az adathalmazok, és az első oszlop a mondathatárokat azonosítja, a másodikban pedig a mondatokat alkotó szavak találhatóak. A TrainNER és Test1NER kibővül egy POS tageket tartalmazó oszloppal, illetve értelemszerűen a TrainNER tartalmazza soronként az elvárt névelemeket, amelyek a feladatleírásban is szerepeltek.

Mivel sem a Test1NER-nél sem a Test2NER-nél nem rendelkeztünk a felismerendő névelemekkel, ezért a TrainNER-t vetettük alaposabb vizsgálat alá. A TrainNER 9000 mondatot tartalmaz és 196645 szót. Összesen 14 névelem szerepel benne, tehát egy, a feladatban ismertetett névelem teljesen hiányzik belőle (I-event). A következő grafikonon vizualizáltuk a névelemek eloszlását az adathalmazban (1. ábra).



1. ábra: Névelemek eloszlása a TrainNER-ben

Az *„O”,* mint *Other* típusú névelemet nem jelenítettük meg az ábrán, mert nagyságrenddel több esetben fordul elő, mint a többi névelem. Összesen 166 610-szer, ami

az adathalmaz közel 85%-a. Tehát, csak *'O'* predikciójával minden sorra már egész magas pontosságot lehetne elérni.

2.2. Névelem felismerő rendszer kiválasztása

A tárgy keretein belül RapidMinert és Python-t használtunk, és mivel személyesen is szeretjük a Python nyelvet ezért házifeladatunkat is elsősorban ezen a nyelven képzeltük el. Internetes keresés után rábukkantunk a SpaCy [1] nevezetű természetes nyelvfeldolgozó megoldásra. A SpaCy többfajta lehetőséget kínál nyelvfeldolgozásához, melyek között a névelem felismerés is szerepel. A jelenlegi szoftvermegoldásokkal összehasonlítva elég jó helyen áll pontosságot és gyorsaságot tekintve, amely a következő ábrán látszódik is (2. ábra):

SYSTEM	YEAR	LANGUAGE	ACCURACY	SPEED (WPS)
spaCy v2.x	2017	Python / Cython	**92.6**	*n/a* ?
spaCy v1.x	2015	Python / Cython	91.8	13,963
ClearNLP	2015	Java	91.7	10,271
CoreNLP	2015	Java	89.6	8,602
MATE	2015	Java	92.5	550
Turbo	2015	C++	92.4	349

2. ábra: Természetes nyelvfeldolgozó megoldások összehasonlítása [2]

Bár az ábra általánosságban hasonlítja össze a nyelvfeldolgozókat, NER megoldás esetén is jónak mondható a SpaCy. További pozitívumot jelentett, hogy a szoftver alkalmas tanítára is az előre megalkotott modellek mellett (1. táblázat):

Név	Méret
en_core_web_sm	*35 MB*
en_core_web_md	115 MB
en_core_web_lg	812 MB

1. táblázat: SpaCy modellek

Ezek alapján nem volt kérdéses, hogy melyik szoftverlehetőséget válasszuk.

2.3. SpaCy tanításának és használatának kipróbálása első megoldás elkészítése

Jobban megvizsgálva a SpaCy-t indokoltnak láttuk, hogy a beépített modellek mellett, saját tanítást is kipróbáljunk, hiszen nekünk specifikus adataink vannak és lehet, hogy általánosságban nem érhetünk el jobb eredményt a modelleknél, de a kötött adathalmaz miatt a pontosság emelése elképzelhető.

Egyetértettünk abban, hogy a vizsgálatok elvégzéséhez válasszuk szét a TrainNER adathalmazt tanító és tesztelő halmazokra, illetve abban is, hogy kizárólag a helyes találatok

aránya nem lesz elegendő mérőszám az eredmények értékeléséhez. Az *Other* entitások nagyon gyakori előfordulása miatt az általános *accuracy* ($1 - \frac{hibásan\ felismert}{összes}$) értéke nem lenne elég kifejező hisz az összes entitásra *'O'* predikálásával már 85% körüli eredményre számíthattunk. Ezért új, sajátos metrikákat dolgoztunk ki, melyeket utalva, az eredeti mérőszámokra *precision*-nek és *recall*-nak neveztünk el. Esetünkben nem bináris osztályozási problémáról van szó, ahol eredetileg értelmezettek az említett metrikák. *Precision* számításnál azt vizsgáltuk, hogy a nem *'O'* entitások közül mennyit talált meg helyesen a használt algoritmus az összes nem *'O'*-nak predikálthoz képest. Tehát azt számoltuk ki, hogy milyen pontosan határozta meg a nem *Other* típusú entitásokat (más megközelítésben: amiket nem *Other*-nek detektáltunk azokat mennyire jól határoztuk meg?) (1).

$$Precision = \frac{eredetileg\ nem\ Other, de\ helyesen\ megtalált\ entitások}{összes\ nem\ Other - nek\ predikált\ entitás} \quad \textbf{(1)}$$

Recall esetén szintén az eredetileg nem *Other,* de helyesen osztályozott típusú entitások számát osztottuk, de ebben az esetben az összes **eredetileg** nem *Other* típusú entitások számával. Tehát megadtuk annak a fedését, hogy a nem *Other* típusokat milyen arányban találta meg a használt algoritmus (2).

$$Recall = \frac{eredetileg\ nem\ Other, de\ helyesen\ megtalált\ entitás}{összes\ eredetileg\ nem\ Other\ entitás} \quad \textbf{(2)}$$

A metrikák meghatározása után már könnyen össze tudtuk hasonlítani a tanítás és a használt modellek használata közötti eredménykülönbségeket.

2.3.1 SpaCy elsőkörös tanítása

A SpaCy tanítását két körben végeztük el. Az első eredmények beküldése előtt, majd később javításként utána is. Elsőkörben alapvetően háromféle paraméterezést próbáltunk ki, melyekről egy összefoglaló táblázat alább látható (2. táblázat).

	First	Second	Third
Iteration	100	128	300
Training Data Size	2000 rows	500 sentences	1000 sentences
Accuracy	87.98%	91.86%	92.03%
Recall Precision	35.17% 34.90%	54.61% 55.95%	55.38% 56.77%

2. táblázat: Elsőkörös tanítási eredmények

A táblázat első oszlopából kiolvasható, hogy 100 iterációval és 2000 sorral próbálkoztunk először (a közel 200 000 lehetséges sorból). A kis adathasználatot az indokolta, hogy a számítási idő már így is elég hosszú, 2.5 óra volt. Látható, hogy az *accuracy* nem sokkal, de meghaladta a 85%-ot. A két számított metrikánk pedig az utolsó sorból olvasható ki: 35%-ban sikerült jól lefedni a nem *Other* entitásokat és közel szintén

35% pontosan. A táblázatból az is látható, hogy a megalkotott mérőszámainkkal hasznos, részletesebb információkat kaptunk a tanításokról, hiszen növekedésük érzékenyebb volt a próbálkozásainkra, mint az *accuracy*.

A további tanításoknál összeállítottuk az adathalmazban megtalálható mondatokat azt gyanítva, hogy a SpaCy tanításánál előnnyel jár a szavak sorrendjének és a mondatok szórendjének az ismerete. A sejtésünket az eredmények teljesen alátámasztották. A második tanításnál 128 iterációt és 500 mondatot használtunk, így 4%-os *accuracy* emelkedést sikerült elérnünk.

A második esetben a futásidő pár perccel volt csak több volt, mint két óra, ezért érdemesnek tartottuk megemelni a paraméterek értékét 300-ra és 1000-re. A harmadik tanítással szintén sikerült javítani a visszamérési értékeken, de már jóval kisebb mértékben (2. táblázat, harmadik oszlopa).

2.3.2 SpaCy modellek kipróbálása

A SpaCy-ban biztosított modellek feltehetőleg lényegesen nagyobb adathalmazon, több erőforrás felhasználásával készültek, mint ami a mi rendelkezésünkre állt. Ebből kifolyólag érdemesnek találtuk kipróbálásukat a házi feladat adathalmazain. Az előre tanított modellek több névelem kategóriát ismertek fel, de nem tartalmazták a kezdő és tartalmazó jelöléseket (*'B', 'I'*).

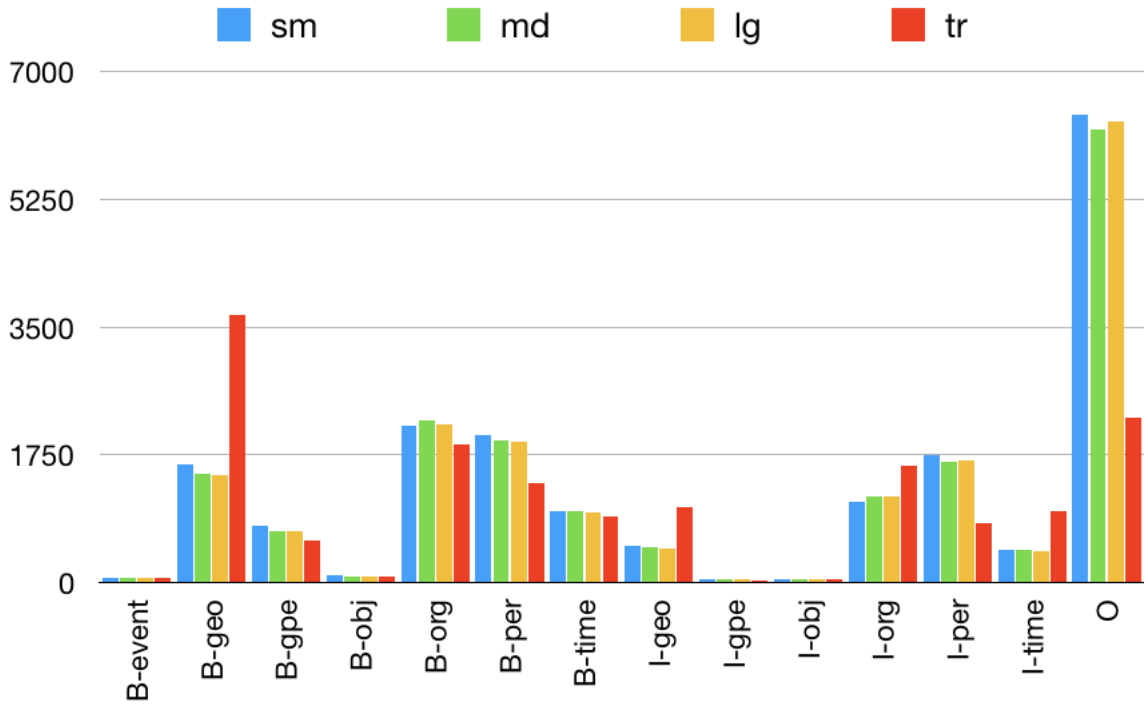
A kezdő és tartalmazott névelemek megalkotása könnyebben megoldható probléma volt a megtalált összetett névelemek darabolásával. A névelem kategóriák pontos megfeleltetése azonban több lehetőséget is tartogatott.

Első próbálkozásként a predikálandó névelemeinkhez kerestük meg név alapján legegyértelműbb párjaikat a SpaCy által biztosítottak között. Ezzel a megoldással azonban csak 87%-os eredményt sikerült elérnünk a TrainNER adathalmaz sorain.

Következő lépésként algoritmikusan döntöttük el, hogy melyik SpaCy kategóriát érdemes megfeleltetnünk a sajátunknak, a következők szerint:

1. Iteratívan veszünk egy saját kategóriát, ami viszonylag sok tévesztéssel járt.
2. A tévesztések között vesszük a leggyakoribb SpaCy kategóriát.
3. A SpaCy kategóriához sorolt szavakhoz meghatározzuk, hogy milyen kategóriába tartoztak eredetileg a saját adathalmazunkban.
4. Kiszámoljuk, hogy milyen változást hozna pontosságban, ha megváltoztatnánk a hozzárendelést (pozitív/negatív hatás)
5. Átsoroljuk a SpaCy kategóriát saját kategóriára pozitív hatás esetén. Majd vesszük a következő saját kategóriát (folytatás az 1. pontnál).

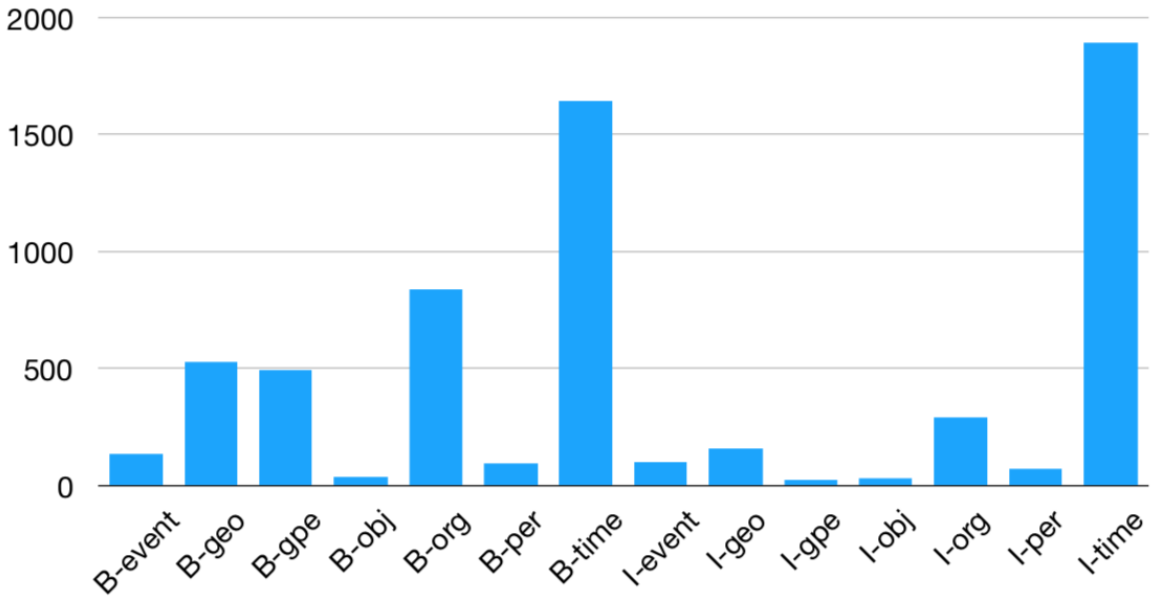
A módszer segítségével hamar kiderült, hogy a SpaCy GPE kategória összes eleme a saját GEO kategóriánknak feleltethető meg, illetve a saját GPE kategóriára a NORP SpaCy kategória illik legjobban. Megvizsgáltuk a *B* és *I* alkategóriák eloszlását is de végül ezeken nem láttuk értelmét javításnak. Az alábbi ábrák (3. ábra, 4. ábra) már a SpaCy és sajátnévelemeink közötti konvertálás után készültek.



3. ábra: A különböző modellek és a tanítás hibáinak darabszáma entitásonként

A fenti ábra (3. ábra) vízszintes tengelyén a TrainNER adathalamaz entitásai szerepelnek. A függőlegesén pedig a kipróbált algoritmusok hibázásainak száma. Például láthatjuk, hogy az előre tanított modellek az O kategorián hibáztak a legtöbbet, míg a tanításnál a B-geo entitás meghatározása volt a legnehezebb.

A fenti 3. ábra is látható eredményeket tovább elemezve az alábbi ábrát készítettük (4. ábra). Leolvasható hogy az 'O' kategóriára a legtöbb hibás predkció a B-time és I-time félre becslésével keletkezett. Felmerült ezen kategóriák elhagyása és általános 'O' címkére konvertálása, azonban méréseink szerint ez nagyobb negatívummal járt volna, mint a megtartásuk.



4. ábra: SpaCy hibás predikciók az O kategóriára

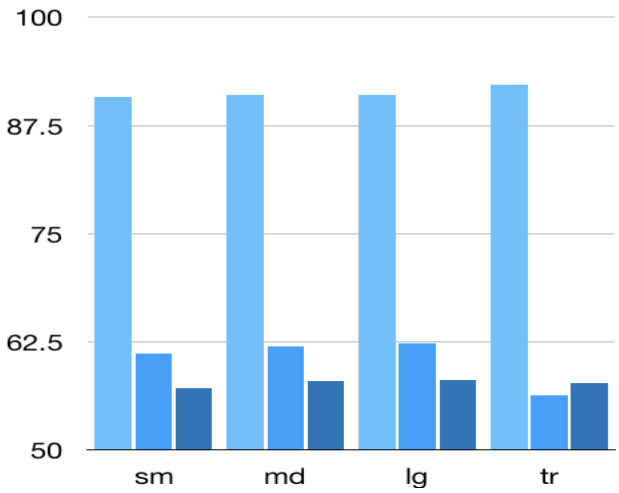
2.3.3 Megoldás kiválasztása és első eredmény beküldés

A modellekkel elért végső eredmények a 3. táblázatban láthatók. A medium (md) és large (lg) modell egyértelműen jobb eredményeket produkált a small (sm) modellnél. A medium és large között már csak a saját metrikáinkkal látszódott a különbség.

A 5. ábra vizuális formában láthatóak a 3. táblázat értékei. A színek megfeleltethetők a táblázat accuracy, recall és precision oszlopainak.

model	accuracy	recall	precision
sm	90.82	61.19	57.19
md	91.03	61.95	57.99
lg	91.03	62.34	58.11
tr	92.03	55.38	56.77

3. táblázat: A különböző modellek és tanítás eredményei



5. ábra: A különböző modellek és a tanítás eredményei

Végül úgy döntöttünk, a *recall* és *precision* metrikákon elért legmagasabb értékek alapján, hogy az *lg (large)* modellt használjuk fel a Test1NER és Test2NER adathalmazok becsléséhez, majd ezt az eredményt adjuk be elsőkörös megoldásként.

Az elért eredményeink Test1NER esetén 91.17%, Test2NER esetén pedig 90,82%.

2.4 Másodikkörös tanítás és az eredmények javítása

A megkapott eredmények után úgy gondoltuk, hogy érdemes lenne saját tanításunkon javítani és a beépített modellek használatát elvetni, hiszen azok kevésbé testreszabhatóak. Két nagyobb tanítást terveztünk. Az első 350 iteráció és 2000 mondat figyelembevételével valósult meg. Sajnos a tanítás futása már meglehetősen lassú volt, ezért éjszaka tudtuk futtatni. A megkétszerezett méretű tanulóállomány nem hozott akkora javulást az előzőekhez képest. 0.15%-kal, 92.18%-ra sikerült emelni az *accuracy*-t, melyhez 56.35% *recall* és 57.98% *precision* társult. Összevetve az elsőkörös legjobb tanítással elmondható, hogy a tanító adatok kétszerezése nem hozott számottevő javulást, a megalkotott metrikáink is csak egy-egy százalékkal emelkedtek. Ennek ellenére úgy véltük, hogy nem

használtuk ki eléggé a tanító adathalmaz méretét és a rendelkezésre álló időnket, ezért érdemes több tanító mintát is bevenni a tanításba.

A végső *36* - *48* órán át tartó tanítást 7200 mondaton 550 iteráció felhasználásával sikerült végrehajtanunk. Ennek lefuttatásához szükség volt apróbb teljesítmény optimalizálásra a tanítás folyamatában, valamint a tanítás és az előfeldolgozási lépések (főképpen mondat alkotás) teljes szétválasztására. Méréseink alapján ez 92.72%-os *accuracy*-t eredményezett. Végül pedig a Test1NER halmazon 92.12%-ot a Test2NER halmazon 92.18%-ot értünk el.

2.5 Összefoglaló

A félév alatt egy érdekes és kihívásokat rejtő házi feladaton gondolkozhattunk. A munka során végig összhangban dolgoztunk és döntéseinket együtt hoztuk meg. A vizsgált adathalmaz elemzése után sikerült egy elterjedt megoldást kipróbálnunk és többféleképpen használnunk névelem felismerés területén. A tanítás és a beépített modellek használata egymáshoz közeli eredményeket hozott, ezért elsőkörben a modellekkel számított predikciókat adtuk be megoldásként. Második körben pedig már a saját legjobb tanításaink közül válogattunk és értünk el javulást.

Hivatkozások

[1] SpaCy szoftvermegoldás, https://spacy.io (2018. 11. 26.)

[2] Természetes nyelvfeldolgozó megoldások összehasonlítása, https://spacy.io/usage/facts-figures#benchmarks (2018.11.26)