

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**](mailto:tmas.szanto@gmail.com)**,** [**krisztianbenda@gmail.com**](mailto: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**

# 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](https://spacy.io): python library, statisztikai modell alapú NER, sokféle kategória támogatott, továbbtanítható saját kategóriákkal
* [Stanford NER is a Named Entity Recognizer](https://nlp.stanford.edu/software/CRF-NER.shtml): Java library, kevés alapból támogatott kategória
* [Named-Entity-Recognition-BLSTM-CNN-CoNLL](https://github.com/mxhofer/Named-Entity-Recognition-BidirectionalLSTM-CNN-CoNLL): implementáció [ehhez a cikkhez](https://arxiv.org/abs/1511.08308), 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

# **Névelem felismerés lehetséges megoldása**

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

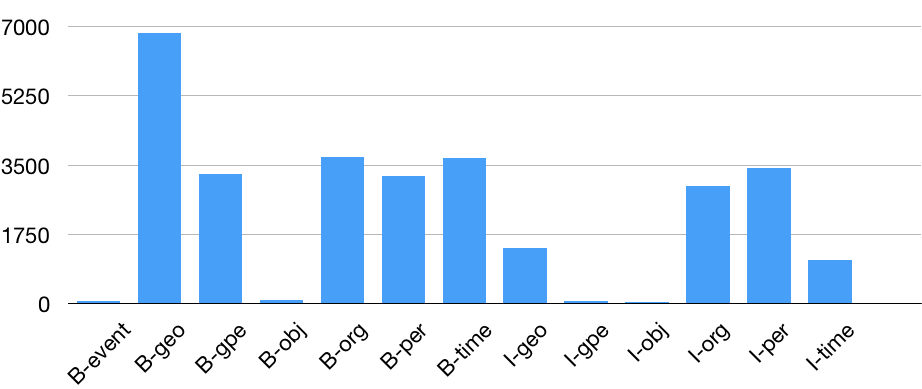
A beszámolónkat is ebben a sorrendben készítettük el, kiegészítve egy összefoglalóval.

## **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).

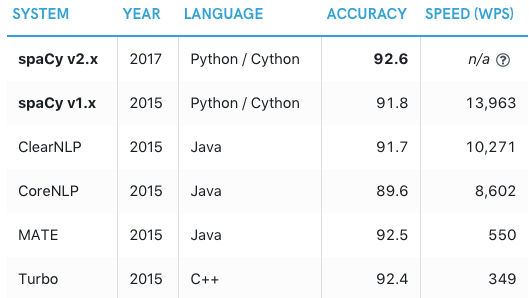


1. ábra: Névelemek eloszlása a TrainNER-ben

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.

## **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):



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.

## **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. Az első megoldás során a két irány kipróbálásához szerencsés volt, hogy a feladat két irányát meg tudtuk osztani magunk között.

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* () 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).

**(1)**

Recallesetén szintén az eredetileg nem *Other,* de helyesen osztályozotttípusú 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).

**(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 által tartalmazott modellek lényegesen nagyobb adathalmaz, feltehetőleg lényeges több erőforrás felhasználásával készültek mint ami nekünk rendelkezésünkre állt. Ezért érdemesnek találtuk ezek kipróbálását a biztosított adathalmazon. Az előre tanított modellek azonban több névelem kategóriát, ám azokat kezdő és tartalmazó állapotok nélkül tartalmaztak.

A kezdő és tartalmazott névelemek megalkotása egyértelműen megoldható volt a megtalált összetett névelemek darabolásával. A névelem kategóriák megfeleltetése azonban nem több lehetőséget is engedett kipróbálni.

Első próbálozásra a prediktálandó névelemeinkhez kerestük meg a név alapján legegyértelműbb párjaikat a SpaCy által biztosítottak között. Ezzel a megoldással azonban csak körülbelül 98%-os eredmény sikerült elérnünk.

Következő lépésként elkezdtünk végig haladni a legtöbbet tévesztett kategóriákon következő módszer szerint:

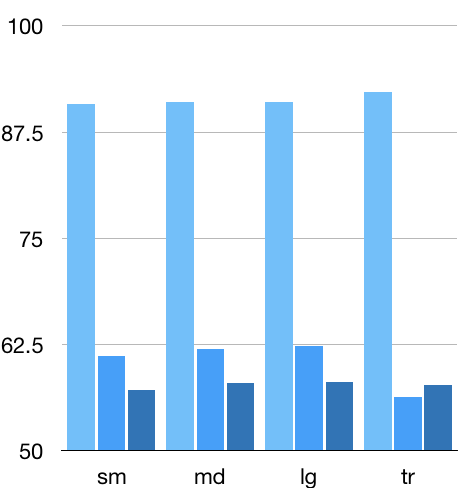
1. Saját kategória, sok tévesztéssel
2. A tévesztettek között mi a leggyakoribb kategória a SpaCy által biztosítottak között
3. A SpaCy kategóriához sorolt szavak milyen kategóriákba tartoznak nálunk
4. Milyen változást hozhat pontosságban ha megváltoztatjuk a hozzárendelést (pozitív/negatív hatás)
5. Átsorolás ha érdemes, folytatás az 1. pontnál ha még van kategória amit érdemes vizsgálnunk.

A módszer segítségével hamar kiderült hogy a SpaCy GPE kategória összes eleme a saját GEO kategóriába tartozik, és a saját GPE kategóriára a NORP passzol a legjobban. Megvizsgáltuk a B és I alkategóriák eloszlását is de végül ilyen téren nem javítottunk. (A 2.3.3 pontban grafikonon demonstrálva is látható a módszer.

A végső eredmények a következő oldali táblázatban láthatók, bár a két nagyobb modell észrevehetően jobb volt a legegyszerűbbnél, a közepes méretű már nem maradt le sokkal a nagyobbhoz képest.

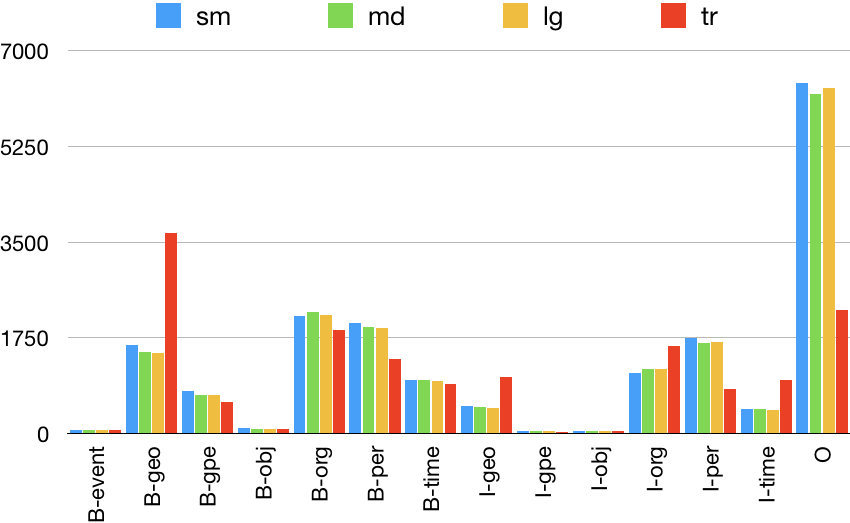
|  |  |  |  |
| --- | --- | --- | --- |
| **model** | **accuracy** | **recall** | **precision** |
| **sm** | 90.82 | 61.19 | 57.19 |
| **md** | 91.03 | 61.95 | 57.99 |
| **lg** | 91.03 | 62.34 | 58.11 |
| **tr** | 92.03 | 55.38 | 56.77 |

3. táblázat: A különböző modellek és tanítás eredményei



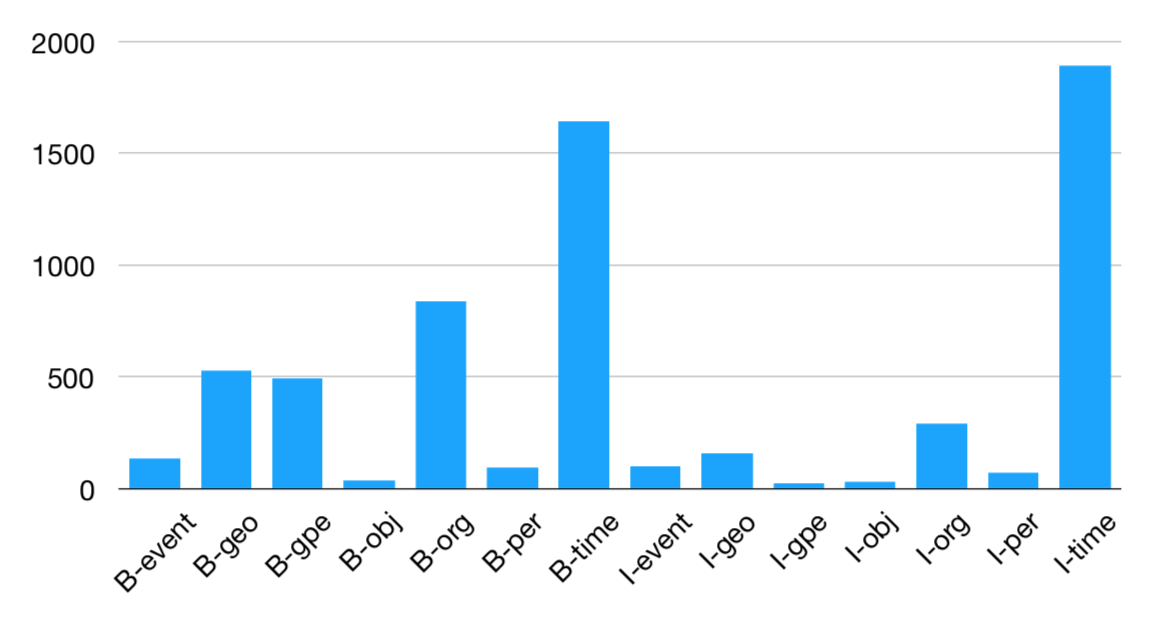
3. ábra: A különböző modellek és a tanítás eredményei

### **2.3.3 Megoldás kiválasztása és első eredmény beküldés**



4. A különböző modellek és a tanítás hibái entitásonként

A fenti ábrán például látszik a hogy az előre tanított modellek az O kategorián hibáztak a legtöbbet. Az alábbi ábráról leolvasható hogy erre a kategóriákra a legtöbb hibás prediction a B-time és I-time katóriák voltak. Felmerült ezen kategóriák elhagyás azonban többet veszítettünk volna a DATE kategória becslésénél mint amit itt nyerni tudtunk volna.



5. SpaCy hibás predikciók az O kategóriára

Végső döntésünk az első beadásra az előre tanított legnagyobb modell predikciói voltak, mivel a saját metrikáink lényegesen magasabb értékei alapján ez tűnt számunkra használhatóbbnak.

Az elért eredményeink: Test1NER 91,17% és Test2NER 90,82%

Még a PPT-ben maradt két grafikon, ha sokat akarunk írni még beletehetjük ☺

... A végén az alábbiakat értük el: Test1NER: 91,17%, Test2NER: 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.

... 7200 mondta 1000 iteráció 500 iterációnál lelőve (ez nem biztos), de a 300.-nál jobb volt az eredmény ...

A végső körülbelül 36-48 órát tartó tanítás alatt 7200 mondaton 550 iterációt sikerült végrehajtanunk. Ennek lefuttatásához szükség volt apróbb teljesítmény optimalizálásokra a tanítás folyamatában, valamint a tanítási és előfeldolgozási lépések (főképpen mondat alkotás) teljes szétválasztására.

## **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, a második körben pedig már a saját legjobb tanításaink közül válogattunk. ...Ha addig megjön az eredmény, akkor ideírhatjuk, ha nem akkor érdemes lenne a hosszú futtatásod leállításakor elértet/legjobbat...

A végső eredményeink:

* Saját leválasztott teszthalmazon visszamérve: 92,72%
* Test1NER: 92,12%
* Test2NER: 92,18%

# 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)