Névfelismerés

Dokumentáció

Szöveg- és webbányászat szorgalmi házi feladat

Készítette:

Kovács László

Tartalomjegyzék

A rendszer célja, funkciói és környezete

Feladatkiírás

A névelem-felismerés (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 entitás több szóból áll, akkor az összes többi I-vel jelölendő (inside), azaz Ievent, I-geo, stb. címkék; így összesen az egyéb (O) címkével együtt 15 osztálycímke adódik.

A programok által biztosítandó funkciók

A program legyen képes tetszőleges korpuszon névfelismerést végezni, és annak eredményét kiírni az előre megadott formátumban egy „csv” kiterjesztésű fájlba. Kritérium még a folyamat reprodukálhatósága is, beleértve az adatok előfeldolgozását illetve esetlegesen a modell tanítását.

A programok környezete

A programok lokálisan futtatható Jupyter Notebook alkalmazások, amelyeknek előfeltétele a szükséges python könyvtárak megléte. Ehhez minden program esetében a dokumentumban leírt telepítési útmutató nyújt segítséget. Ezen felül az Azure névfelismerő szolgáltatását használó programnak szüksége van internet elérésre, illetve az online szolgáltatás elérhetőségére. A dokumentumban hivatkozott és bemutatott alkalmazás a saját Azure fiókomban elérhető szolgáltatást használja, a programkódba statikusan megadott elérési útvonallal és API kulccsal. Ezen adatok nem publikusak.

A környezet kialakítása

A programok futtatásához szükséges valamilyen program, ami képes a Jupyter Notebook-ok futtatására. Ehhez egy megoldás az alábbi linken található részletes leírással. A fejlesztéshez a Visual Studio Code fejlesztői környezetet használtam, ami támogatja a Jupyter Notebook alkalmazások futtatását.

A Jupyter Notebook installálását és elindítását követően a böngészőben tudunk a fájlrendszerünkben navigálni. A programokat és a feladathoz tartozó adatokat az alábbi linkről lehet letölteni:
https://github.com/Makrenos/szwb

Ezek után a „JNH1DH” kezdetű „.ipynb” kiterjesztésű fájlokat megnyitva futtathatóak az alkalmazások.

A programok futtatható blokkokat tartalmaznak, melyeket sorrendben kell lefuttatni ahhoz, hogy a megfelelő eredményt kapjuk. A programok felépítése a következő struktúrát követik:

- Szükséges könyvtárak betöltése
- Adatok betöltése
- Adatok előkészítése
- Adatok értékelése
- Adatok megfelelő formátumra hozása
- Adatok kiírása fájlba

Első megoldás

Az első megoldás a „JNH1DH_Senford” Jupyter Notebook fájlban található. Ez a program a Stanford névfelismerő modelljét használja, amihez szükség van a futtatható java fájlra, ami a projektben rendelkezésre áll, így azt nem kell külön letölteni.

Szükséges könyvtárak

A program működéséhez szükséges az alábbi könyvtárak legfrisseb verziójának installálása:

- Pandas
- Numpy
- NLTK

A program leírása

A program első futtatható blokkja tartalmazza a szükséges könyvtárak betöltését.

A második blokkban található a teszt adatok betöltése, és megfelelő formátumra hozása. Az első sor el lett távolítva, és az oszlopok nevei fel lettek címkézve.

A harmadik blokkban a NER modell betöltését és a szavak címkézést végző funkció található, ami bemenetnek egy listát vár, ami a címkézendő szavak listáját várja, kimenete pedig a címkék.

A negyedik blokk a BIO címkézést megvalósító funkciót tartalmazza. Bemenetén egy listát vár, ami a címkéket tartalmazza, kimenete a címkék BIO előtaggal ellátva. Működésének lényege, hogy végig iterál a listán, számon tartva az előző címke milyenségét, és összehasonlítások alapján dönti el, hogy az aktuális címke milyen BIO előtagot kapjon.

Az ötödik blokkban található a test adatok felcímkézése a modell segítségével. Először a program kigyűjti azon szavakat tartalmazó cellákat, amiknek értéke definiálva van, azaz nem üres. Ezek után a szavak listáját átadja a NER modellnek címkézésre, aminek visszatérési értékét a BIO címkéző veszi át és módosítja azt. Az eredménnyel a teszt adathalmaz egy új „Predicted” nevű oszlopát tölti fel a program.

A hatodik és hetedik blokk funkciója megegyezik, az egyetlen különbség, hogy míg a hatodik blokk az első teszt adathalmaz formázást és kimentését végzi addig a hetedik blokk a második teszthalmaz formázását és fájlba történő kiírását végzi. A blokkok végig iterálnak az újonnan feltöltött „Predicted” oszlopon, és az előre megadott elvárt formátumra hozzák a címkéket, illetve a feladat szempontjából nem releváns címkéket felcserélik „O” címkére.

A blokkok megadott sorrendben való futtatása után a gyökér könyvtárban megtalálhatóak a „Submission1_Stanford.csv” és „Submission2_Stanford.csv” fájlok,

amelyek tartalmazzák a teszt adathalmaz minden sorát és oszlopát kiegészítve a beazonosított névelemek címkéjével a „Predicted” oszlopban.

Második megoldás

Az második megoldás a „JNH1DH_Azure” Jupyter Notebook fájlban található. Ez a program az Azure névfelismerő modelljét használja, amihez szükség van az online szolgáltatás elérésére.

Szükséges könyvtárak

A program működéséhez szükséges az alábbi könyvtárak legfrisseb verziójának installálása:

- Pandas
- Azure Text Analytics client library for Python

A program leírása

A program első futtatható blokkja tartalmazza a szükséges könyvtárak betöltését, a teszt adatok betöltése, és megfelelő formátumra hozása. Az adat táblák első sora el lett távolítva, és az oszlopok nevei fel lettek címkézve.

A második blokkban egy segédfüggvény található, ami bemenetnek egy Pandas-os „data frame”-et vár, olyat, amibe a teszt adatok be lettek töltve. Kimenete egy lista, ami tartalmazza az adathalmazban található összes mondatot.

A harmadik blokk az Azure szolgáltatáshoz való kapcsolódást megvalósító klienst inicializálja.

Az negyedik blokkban segédfüggvények találhatóak. Az első függvény címkéket tartalmazó listát vár, amiket BIO címke előtagokkal lát el. A második segédfüggvény valósítja meg az adatok küldését az Azure szolgáltatásnak, a válasz fogadását és feldolgozását. A függvény bemenetnek egy „data frame”-et vár, kimenetként pedig egy listát ad, ami az összes beazonosított entitás címkéjét tartalmazza már vissza alakítva szavakra. A harmadik, egyben utolsó függvény az adat előfeldolgozását végzi. Bementnek egy „data frame”-et vár, amit előformáz, majd szegmentál. Kimenetnek egy listákat tartalmazó listát ad.

A szegmentálásra és a beazonosított entitások címkéjének utó feldolgozására az Azure szolgáltatás sajátossága miatt van szükség. Az Azure ingyenes előfizetése csak havi 5000 lekérdezést tesz lehetővé. Ha minden mondatot egyenként küldene a program akkor már az első teszt adathalmaz kiértékelése közel kerülne a limithez. Ezért van szükség több mondat egyszerre való elküldésére. Viszont van egy felső karakter limitje is az egyszerre elküldhető karaktereknek, ezért ilyen „körülményes” az adatok előfeldolgozása.

A válaszban érkező névelem entitások címkézése szintén sok utómunkát igényel, mivel a mondatban megtalált névelem entitásokat a válasz összekonkatenálva, egyként küldi vissza, viszont a végeredményben szavankénti bontás az elvárt. Ezen felül ha egy

névelemet több címkével is ellátott, különböző konfidencia értékekkel, akkor ezeket az eseteket is kezelni kell, hogy ne legyen ütközés az eredmények beillesztésénél. Ezen felül néhány egyedi esetben (ahol egy személy neve aposztrófot tartalmaz) hibás kiértékelés születik, ezt is egyedien kezeli a program.

Az ötödik blokk elején regexek találhatóak. Ezekre a címkék elvárt alakra hozásához van szükség. Az Azure által adott címkék nem egyeznek a házifeladatban elvárt címkékkel, ezért végig kell iterálni az egész adathalmazon, és ha egy keresett kategóriára egyezést mutat a regex, akkor azt le kell cserélni az elvárta. Ha nincs egyezés, azaz nem ekresett kategória címke jön akkor azt le cseréli a program „O” címkére. Ezt a feladatot valósítja meg a „prediction_formatting” függvény, melynek bemente egy „data frame”. Ebben a blokkban található még meg a predikciókat beillesztő függvény is „insertPred” néven, ami bemenetnek szintén egy „data frame”-et vár, majd azon és a címkéken egyszerre végig iterálva beilleszti a predikciót, ha annak szövege egyezik a teszt adathalmaz megfelelő sorában található szóval. Az utolsó függvénye a blokknak a „tag_all_words”, ami bemenetnek vár egy „data frame”-et, majd ezen adathalmaz predikciós címkéire meghívja a BIO tegelést végző függvényt.

A hatodik, hetedik, nyolcadik, illetve kilencedik blokkban az előbbiekben taglalt függvények megfelelő sorrendben történő meghívása látható a két teszt adathalmazra, majd azoknak a megfelelő fájlba történő írását megvalósító függvényhívás.

Harmadik megoldás

A harmadik megoldás a „JNH1DH_Spacy” Jupyter Notebook fájlban található. Ez a program a Spacy általános nyelvi feldolgozó könyvtárait használva épít egy NER modellt üres nyelvi modellre a „Train” adathalmaz segítségével, majd ezzel a modellel végez névfelismerést a teszt adathalmazon.

Szükséges könyvtárak

A program működéséhez szükséges az alábbi könyvtárak legfrisseb verziójának installálása:

- Pandas
- Spacy
- Plac

A program leírása

A program első futtatható blokkja tartalmazza a szükséges könyvtárak betöltését, a teszt adatok betöltése, és megfelelő formátumra hozása. Az adat táblák első sora el lett távolítva, és az oszlopok nevei fel lettek címkézve.

A második blokkban található a modell tanítását és kimentését végző metódus. Ez a függvény megtalálható a Spacy hivatalos honlapján több helyen is, ezen kód elérésnek forrás címét a program meghivatkozza. Egyedüli változtatás, hogy nem az egész tanító halmazon végzi a kód a tanítást, hanem annak utolsó sorát kihagyja. Ennek oka az, hogy egyéb esetben hibába ütközik a tanítás. Ennek miértjét nem sikerült kideríteni.

Szeretném itt megjegyezni, hogy a tanítás újbóli futtatása nagyon lassú lenne, mivel az a processzor számítási egységeit terheli. A Spacy alapján lenne lehetőség ezt a számítást a videókártyára rábízni, ami szignifikánsan gyorsítaná az eljárást, ám több napi kísérletezés és hiba bogarászás után sem sikerült ezt életre kelteni. Windows platformon készült a megoldás, így feltételezem, hogy ez valamilyen Windows specifikus hátrányosság.

Szintén itt jegyezném meg, hogy a „offsets_from_biluo_tags” GoldParse osztály függvénye nem megfelelően viselkedik „for-each” ciklusban. Ez nagyon sok kellemetlenséget okozott. A problémával kapcsolatban a Spacy hivatalos GitHub platformján „Issue”-t nyitottam, a hiba bejelentését megtettem.

A harmadik blokkban az első függvény a BIO címkézés BILOU címkézésre való átalakítását végzi. Erre azért van szükség, mert a tanító adathalmaz BIO címkézéssel tartalmazza a névelem címkéket, viszont ahhoz, hogy a Spacy NER modelljét tanítani lehessen szükség van a névelem címkék BILOU címkézésre. A második függvény a BIO címkézést végzi a megtalált, BIO címkézés nélküli névelem predikciókra. Mindkét függvény névelem címkéket vár listában bemenetnek és azt is ad visszatérési értékként.

A negyedik blokk segédfüggvényeket tartalmaz az adathalmazok elő, illetve utó feldolgozásához. Ezekre nem térek ki mind, nagy részük az előző programban részletesen tárgyalva lett. A függvényeik nevei és logikája megegyezik.

A „get_all_entities” függvény a betanított modellt használva mondatonként nyeri ki a névelem kategóriákat, majd ezeket egy listába fűzve adja visszatérési értékként.

A „substract_list” függvény a névelem címkék „szétszedését” végzi. Erre azért van szükség, mert a névelemeket listák a listában szerkezetben adja vissza a névelem felismerő, a programnak viszont egy listába fűzve van szüksége a címkékre.

A „substract_entites” a felismert névelemek felbontását végzi. Ugyan az a jelenség mint az Azure-os megoldásnál, ha egy beazonosított névelem több szóból áll, akkor azt szavankénti bontásra kell hozni, hogy össze lehessen fésülni az eredeti teszt adathalmazzal.

A „getAllTrainData” függvény a tanító adathalmaz megfelelő formátumra hozását végzi a Spacy modell tanításához. Ehhez szükség van a tanító adathalmaz mondatainak kinyerésére, majd azok alábbi képen látható formátumra hozására.

```
TRAIN_DATA = [
        ("Uber blew through $1 million a week", {"entities": [(0, 4, "ORG")]}),
        ("Google rebrands its business apps", {"entities": [(0, 6, "ORG")]})]
```

1. ábra *Spacy tanító adathalmaz elvárt formátuma*

Az ábrán látható először a mondat, majd a mondatban megtalálható névelemek, ellátva a mondatban található helyükkel. A mondatban elfoglalt hely a kezdő, majd záró karakter pozíciójaként értendő. Ezt követi a névelem címkéje.

A fent említett formátum előállítását segítő Spacy könyvtári funkció a „offsets_from_biluo_tags”, ami BILOU címkével ellátott névelem kategóriákat megkeresi a mondatban, és megadja a hozzájuk tartozó „offset”-et.

Az ötödik blokkban található a tanító adathalmaz megfelelő formátumra hozását meghívó szekvencia.

A hatodik, hetedik, nyolcadik, illetve kilencedik blokkban a segédfüggvények megfelelő sorrendben történő meghívása látható a két teszt adathalmazra, majd azok fájlba írása.

A program készítése során felhasznált eszközök

- Python [1]
    - Felhasználás: programozási nyelv
- Jupyter Notebook [2]
    - Felhasználás: interaktív programozási környezet
- Stanford Named Entity Recognizer [3]
    - Felhasználás: névelem felismerés
- Azure Cognitive Services [4]
    - Felhasználás: névelem felismerés
- Spacy [5]
    - Felhasználás: névelem felismerés
- GitHub [6]
    - Felhasználás: verziókezelés, projekt online elérhetősége
- Visual Studio Code [7]
    - Felhasználás: fejlesztői környezet
- Microsoft Word
    - Felhasználás: dokumentum szerkesztés

Hivatkozások

Hiba! A hivatkozási forrás nem található. Python
https://www.python.org/

[2] Jupyter Notebook
https://jupyter.org/

[3] Stanford Named Entity Recognizer
https://nlp.stanford.edu/software/CRF-NER.html

[4] Azure Cognitive Services
https://docs.microsoft.com/en-us/azure/cognitive-services/

[5] Spacy
https://spacy.io/

[6] GitHub
https://github.com/

[7] Visual Studio Code
https://code.visualstudio.com/