

# Text Enrichment

## Dokumentáció

Text Enrichment

Mesterséges intelligencia alapú szöveginformáció gazdagító  
szoftvermegoldás

Szoftverarchitektúra tárgy házi feladat

Készítették

Benda Krisztián  
Szántó Tamás

# Tartalomjegyzék

## Text Enrichment Rendszerterv

<b>TARTALOMJEGYZÉK.....</b>	<b>2</b>
<b>A RENDSZER CÉLJA, FUNKCIÓI ÉS KÖRNYEZETE.....</b>	<b>3</b>
<b>FELADATKIÍRÁS .....</b>	<b>3</b>
<b>A RENDSZER ÁLTAL BIZTOSÍTANDÓ TIPIKUS FUNKCIÓK .....</b>	<b>3</b>
<i>Felhasználói felület, kliens alkalmazás.....</i>	<i>3</i>
<i>Szövegfeldolgozó szolgáltatás.....</i>	<i>3</i>
NER motor: .....	3
Webszerver: .....	3
<b>A PROGRAM KÖRNYEZETE .....</b>	<b>4</b>
<b>MEGVALÓSÍTÁS.....</b>	<b>5</b>
<b>ARCHITEKTÚRA.....</b>	<b>5</b>
<i>NER Server .....</i>	<i>6</i>
NER Engine.....	6
NER Server Rest API .....	6
<i>WebServer.....</i>	<i>7</i>
Business Logic.....	7
WebServer Rest API.....	9
<i>User Interface .....</i>	<i>11</i>
<b>TELEPÍTÉSI LEÍRÁS.....</b>	<b>15</b>
<b>A PROGRAM KÉSZÍTÉSE SORÁN FELHASZNÁLT ESZKÖZÖK .....</b>	<b>16</b>
<b>ÖSSZEFOGLALÁS .....</b>	<b>17</b>
<b>TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK.....</b>	<b>18</b>
<b>HIVATKOZÁSOK.....</b>	<b>19</b>

# A rendszer célja, funkciói és környezete

## Feladatkiírás

Az elkészítendő megoldás egy szövegfeldolgozó szolgáltatást megvalósító szerveralkalmazás és a szolgáltatásra épülő kliensalkalmazás. A szolgáltatás szöveges tartalmat képes feldolgozni, és az eredményét visszajelezni.

A szöveg feldolgozása során az algoritmus elemzi a bemeneti szavak jelentését, szófaját és információ tartalmát. Az elemzés eredménye a szavak jelentéséhez kapcsolódó információk feltárása és új információkkal való ellátása. A felhasználó a kliensalkalmazás segítségével interakcióba léphet a szolgáltatással és felhasználhatja annak eredményeit.

## A rendszer által biztosítandó tipikus funkciók

Az elkészített rendszer 3 modulra osztható. Egy felhasználói felületet megvalósító kliensalkalmazásra, egy névelem felismerő (NER) motorra és a kettő kommunikációját összekötő és információkat gyűjtő webszerverre. A webszerver és a NER motor külön szolgáltatást kínál a klienseknek, ezért a biztosítandó funkciókat is elkülönülten mutatjuk be.

### *Felhasználói felület, kliens alkalmazás*

- Angolnyelvű szöveg bevitelének támogatása másolással, illetve gépeléssel
- A megadott szöveg feldolgozása során szükséges fogalmak beállítása
- A szövegfeldolgozó szolgáltatás használata a bevitt szövegen
- A feldolgozás eredményének megjelenítése, megtalált fogalmak kiemelése
- A szövegfeldolgozás során megtalált hivatkozások, linkek elérhetővé tétele, megjelenítése
- Fogalmakat összesítő nézet megjelenítése

### *Szövegfeldolgozó szolgáltatás*

NER motor:

- Megadott szöveg feldolgozása
- Szavakhoz kapcsolódó információk felismerése és entitásokhoz társítása: névelemek felismerése (*Named Entity Recognition, NER*)
  - Felismerendő entitások: dátum, esemény, földrajzi entitás, geopolitikai entitási, idő, műtárgy, személy, szervezet

Webszerver:

- A NER motor által felismert fogalmakhoz információk társítása: “szöveg gazdagítása”

- A megtalált fogalmak csoportosítása és lekérdezhetővé tétele
- A fogalmak és hozzájuk talált metainformációk alapján hivatkozások keresése
- Az egyes entitásokhoz a következő típusú hivatkozások biztosítása:
  - Dátum: adott időpontban esemény létrehozásra hivatkozás Google Calendar-ban
  - Esemény: Wikipédia oldal
  - Földrajzi entitás: földrajzi hely azonosítására szolgáló Google Maps link
  - Geopolitikai entitás: kapcsolódó földrajzi hely azonosítására szolgáló Google Maps link
  - Idő: adott időpontban esemény létrehozásra hivatkozás Google Calendar-ban
  - Műtárgy: képre történő hivatkozás
  - Személy: Wikipédia oldal
  - Szervezet: Wikipédia oldal
- A NER motorral megtalált névelemek elérhetővé tétele

## A program környezete

A programot vékonykliens alkalmazásként készítettük el. Annak érdekében, hogy több operációs rendszerben (Windows, Mac OS stb.) is futtatható legyen, platformfüggetlen megoldásokat választottunk. A programot több nyelven fejlesztettük. A háttérben működő szolgáltatást Python3 nyelven implementáltuk, míg a felhasználói felület kialakítását JavaScripttel oldottuk meg.

A felhasználói felület megjelenítéséhez szükség van egy előre telepített böngészőre. A szövegfeldolgozó szolgáltatás eléréséhez Python 3 szükséges és egyéb ingyenesen letölthető Python csomagok. Részletesebben lásd a Telepítési leírást.

# Megvalósítás

Az alkalmazást a feladatkiírásnak megfelelően egy többretegű alkalmazásként készítettük el. A rétegek elkülönülnek. Az alkalmazás felhasználói felületre, webserverre és szintén egy webservice formájában elkészített NER motorra osztható. A szolgáltatások különállóan is működtethetők és HTTP metódusokkal elérhetők.

Az általunk elkészített programot Text Enrichment névre kereszteltük, utalva arra, hogy a megadott szöveg információtartalmát többszörösére növeljük új információkkal társításával.

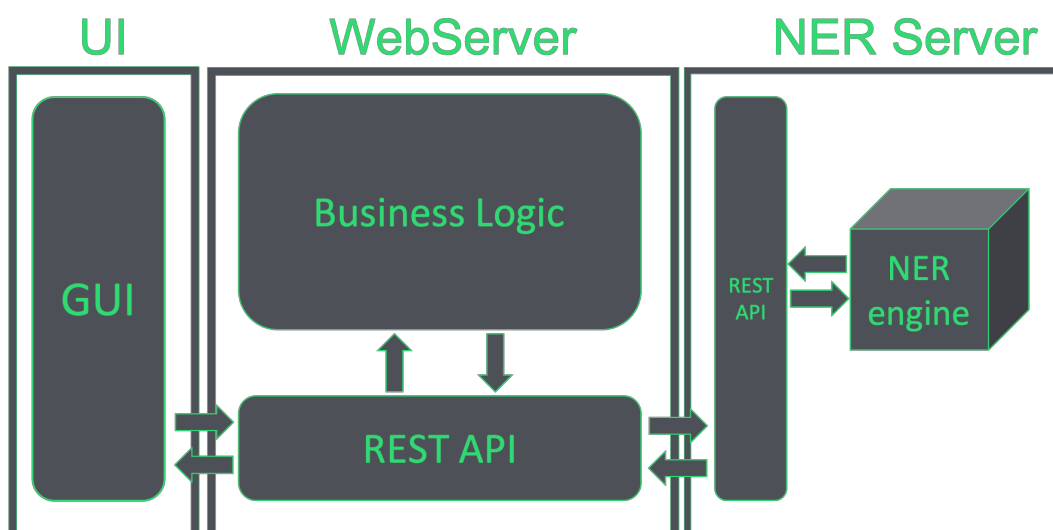
A fejezetben áttekintést adunk a program architektúrájáról, bemutatjuk az egyes komponensek feladatait, különálló használatuknak lehetőségeit, illetve együttes kommunikációjukról is szót ejtünk.

## Architektúra

A Text Enrichment architektúrája 3 különálló nagyobb modulra bontható. A webservice és a NER szolgáltatás további kisebb komponensekre osztható.

- Felhasználói felület (*User Interface, UI*)
  - Grafikus felhasználói felület (*Graphical User Interface, GUI*)
- WebServer
  - Rest API (*REpresentational State Transfer*)
  - Üzleti logika (*Buisness Logic, BL*)
- NER Server
  - Rest API (*REpresentational State Transfer*)
  - Feldolgozó motor

A modulok és komponensek kommunikációját mutatja be az 1. ábra. A fejezet hátralévő részében az egyes komponensek feladatait és felelősségeit ismertetjük.



1. ábra: A szoftver architektúrája

## NER Server

**Célja:** Névelem felismerő algoritmus biztosítása és futtatása. Elérhetőség megteremtése REST interfészen keresztül.

Ahogy már ismertettük a NER Server két részre osztható. NER algoritmust futtató motorra és a NER Server REST interfészét megvalósító komponenesre. A NER algoritmus futtatásához a SpaCy névelem felismerő rendszer [1] angol nyelvű modelljeit [2] használtuk.

### NER Engine

Azért döntöttünk a SpaCy használata mellett mert jelenleg a legjobb eredményeket érte el [3] az aktuális problémánk (NER) terén, valamint úgy tűnt, hogy a szolgáltatásunk megvalósításához ez a legalkalmasabb. Többfajta modellt is biztosít, ami bővíti a szoftver használhatóságát.

A SpaCy három angol nyelvű modelljét támogatjuk az alkalmazásunkban:

- *en\_core\_web\_sm*: 35 MB, kevésbé pontos, ellenben gyorsan betölthető [4]
- *en\_core\_web\_md*: 115 MB, pontosabb, nagyobb mérete miatt lassabban betölthető [5]
- *en\_core\_web\_lg*: 812 MB, elég pontos, de betöltése lassabb az előzőknél [6]

A NER engine futtatásához szükséges letölteni a SpaCy keretrendszert illetve az említett modelleket. Bővebben lásd a Text Enrichment telepítési leírást.

A NER engine feladata a SpaCy algoritmus meghívása és a szövegfeldolgozás elvégzése, majd az eredmény kezelése.

### NER Server Rest API

A NER Server Rest API biztosítja a szerver a NER engine futtatásához, illetve további szerepe, hogy HTTP POST metódust biztosítson a NER engine eléréséhez. A motor ezáltal egy szolgáltatásként érhető el és felhasználható más architektúrában is.

A POST metódus a következő struktúrájú JSON objektumot várja a <http://127.0.0.1:5001/text-processing/ner> címen (a port állítható a szerver indítási paraméterével):

```
{
  "id": "process ID",
  "text": "text which will be processed by the engine",
  "endpoint": "an endpoint which can accept the process results"
}
```

Ahogy a struktúrában is látszódik az „*endpoint*” kifejezés, a REST API feldolgozás végén meghív egy *endpoint*-ot szintén POST metódussal. Tehát a NER Servert szolgáltatásként használó kliensnek felkell erre készülnie. A visszakapott eredmény úgynevezett entitásokat tartalmaz az alábbi struktúrában:

```
{
  "id": "process ID",
  "entities": [
    "found expression",
```

```

        "label",
        <starting_char>,
        <ending_char>
    ],
    [
        <other entity>
    ]
}

```

A struktúra elemeinek jelentése a következő:

- *process ID*: az az ID, amivel az eredeti szöveg publikálásra került a NER Server felé. Esetünkben, ahogy később látni fogjuk ez *UUID*-kat jelent.
- *found expression*: a feldolgozott szövegben található kifejezés, amit az algoritmus talált.
- *label*: a megtalált kifejezéshez tartozó címke. A SpaCy a következő címkéket tudja megkülönböztetni: *PERSON, NORP, FAC, ORG, GPE, LOC, PRODUCT, EVENT, WORK\_OF\_ART, LAW, LANGUAGE, DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL*. Ezek mind publikálásra kerülnek a megadott struktúrában.
- *starting\_char*: egy egész szám. A megtalált kifejezés kezdetét határozza meg a feldolgozott szövegben.
- *ending\_char*: egy egész szám. A megtalált kifejezés végét határozza meg a feldolgozott szövegben.
- *other entity*: az első mintának megfelelő két szöveges és két egész szám mezőt tartalmazó entitás, illetve további tetszőleges számú entitást jelöl.

A NER Server REST API-jának futtatásához a következő Python 3 csomagokra van szükség: *argparse* [7], *threading* [8], *requests* [9], *simplejson* [10], *flask* [11], *werkzeug* [12]

## WebServer

**Célja:** Megvalósítja a szöveg gazdagítását, illetve kommunikál a UI-al és a NER Serverrel.

A WebServer feladata kettős. Egyrészt kommunikálni a környező modulokkal (NER Server, UI), illetve az üzleti logika megvalósítása is a felelősségébe tartozik. A WebServer fel van készítve a NER Server kommunikációjára, biztosít számára egy *endpoint*-ot, ahova az eredményeket várja.

## Business Logic

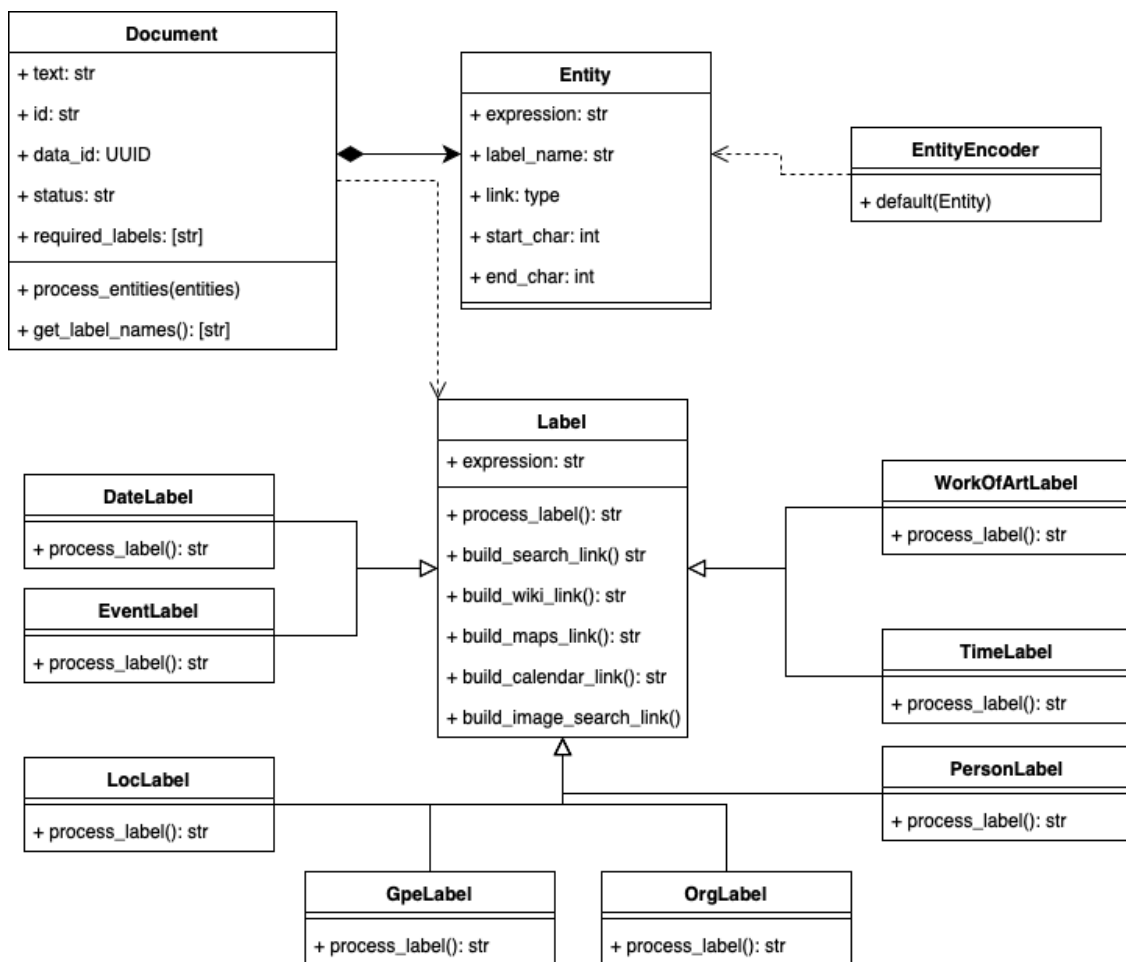
A NER Servertől visszakapott címkék alapján az Üzleti Logika különböző hasznos információkat állít össze a kifejezésekhez. Melyek a következők címkékként:

- *DATE*: a megadott dátum alapján Google Calendar esemény létrehozására link. Az eseményt 1 órára tervezve. Amennyiben nem hozható létre ilyen esemény, akkor nem adunk vissza értelmezhető linket, csak egy „*NOT\_SUPPORTED*” szöveges információt (Ilyen például a „last week” kifejezéssel érhető el).
- *EVENT*: angol nyelvű Wikipédia hivatkozás keresése alapértelmezetten. Amennyiben nem található ilyen oldal, akkor Google keresés link az adott kifejezése.
- *GPE*: geopolitikai entitásra mutató Google Maps link. Amennyiben nem található ilyen, akkor Wikipédia hivatkozás, ha ilyet sem sikerül találni, akkor Google keresés link.

- *LOC*: földrajzi entitásra mutató Google Maps link. Amennyiben nem található ilyen, akkor Wikipédia hivatkozás, ha ilyet sem sikerül találni, akkor Google keresés link.
- *ORG*: angol nyelvű Wikipédia hivatkozás keresése alapértelmezetten. Amennyiben nem található ilyen oldal, akkor Google keresés link az adott kifejezése.
- *PERSON*: angol nyelvű Wikipédia hivatkozás keresése alapértelmezetten. Amennyiben nem található ilyen oldal, akkor Google keresés link az adott kifejezése.
- *TIME*: a megadott dátum alapján Google Calendar esemény létrehozására link. Az eseményt 1 órára tervezve. Amennyiben nem hozható létre ilyen esemény, akkor nem adunk vissza eredményt.

Fontos, hogy a NER Server többféle entitást is megkülönböztet, de mi csak a fentiekkel foglalkozunk. Tehát a szöveg gazdagítását a benne megtalálható kifejezésekhez különböző információk társításával érjük el.

A Business Logic az alábbi osztálydiagrammon látható osztályokat kezeli.



2. ábra: Business Logic által használt osztályok

Egy Document objektum egy a WebServer-nek feladott szöveget reprezentál. Szöveget feladni POST metódussal lehet. A feladandó JSON-re annyi a megkötés, hogy tartalmazzon egy *“text”* nevű kulcsot és hozzá értéket. Az így megadott szövegből a szerver egy *Document* objektumot állít elő, amelyet ellát egy ID-val (*doc-xxxxxx* formában). A *Document* objektum feldolgozásának lépéseit követi a *status* tagváltozó. A feldolgozáshoz meg lehet adni, hogy



milyen típusú kifejezéseket válogassunk ki. Ezeket a típusokat tárolja a *required\_labels* tagváltozó. A *Document* létrehozása után feladjuk egy külön UUID-val ellátott *Document* szövegét a NER Servernek, hogy végezze el a névelemek felismerését.

Az *id* és *data\_id* elkülönítésére azért van szükség, hogy a WebServer szolgáltatását használó kliens ne férjen hozzá a NER Server által kínált szolgáltatás eredményeihez.

A NER Server a kifejezésekhez *Label*-öket társít és *Entity*-ket küld vissza. Az adott típusú objektumok csak a WebServer oldalán jönnek létre, addig csak egyszerű szöveg formjában jelennek meg. A visszaadott *Entity*-ket eltároljuk a *Document*-ben is, amelyekhez a megkapott információkon kívül a fent bemutatott linkeket állítjuk össze és tároljuk el a *link* tagváltozóban. A *Label*-ből származó osztályok a hivatkozás összeállításánál játszanak szerepet, ahol a különböző linkek összeállítását a *process\_label* függvényben vezéreljük.

A Business Logic futtatásához szükséges telepíteni az alábbi Python 3 csomagokat: *simplejson* [10], *uuid* [13], *googlemaps* [14], *wikipedia* [15], *datetime* [16], *dateutil* [17], *unicode* [18], *random* [19].

## WebServer Rest API

A WebServer Rest API komponense végzi el a szükséges kommunikáció támogatását. A feldolgozást megindítani a <http://127.0.0.1:5000/text-enrichment/new-doc-ra> küldött POST metódussal lehet. A feladott JSON-nek csak egy *"text"* kulcs-érték párral kell rendelkeznie. A Rest API ezután meghívja a NER Server *endpoint*-ját a <http://127.0.0.1:5001/text-processing/ner> címen és feladja a szükséges struktúrájú JSON-t. Miután a NER Szerver végzett, a Web Server-nek küldi vissza az eredményt a <http://127.0.0.1:5000/text-enrichment/processed-entities> címen.

A Business Logic feldolgozása után az eredményeket a következő címekre küldött GET metódussal lehet elérni.

- [http://127.0.0.1:5001/text-enrichment/<doc\\_id>/results](http://127.0.0.1:5001/text-enrichment/<doc_id>/results): A teljes eredményt adja vissza adott dokumentum azonosítóhoz JSON-be ágyazva:

```
{
  "status": "processed",
  "entities": "enriched text with links",
  "text": "original text"
}
```

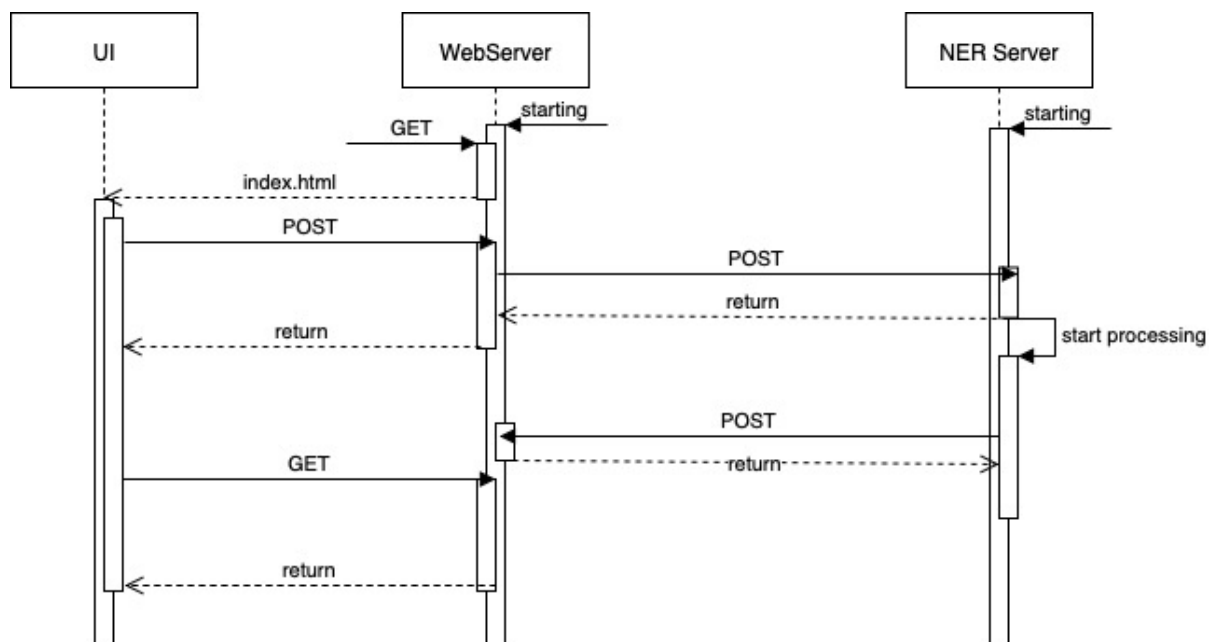
- [http://127.0.0.1:5001/text-enrichment/<doc\\_id>/labels](http://127.0.0.1:5001/text-enrichment/<doc_id>/labels): adott dokumentum azonosítóhoz a megtalált *label*-öket adja vissza. A következő példában látható módon:

```
{
  "labels": [
    "EVENT",
    "PERSON",
    "TIME"
  ]
}
```

- [http://127.0.0.1:5001/text-enrichment/<doc\\_id>/summary](http://127.0.0.1:5001/text-enrichment/<doc_id>/summary): Egy összesített eredményt ad vissza a dokumentumról. A válaszban a különböző *label*-ökhöz talált kifejezések csoportosítva jelennek meg. Az alábbi JSON struktúra erre mutat egy valós példát:

```
{
  "summary": {
    "EVENT": [
      {
        "expression": "the Olympic Games",
        "start_char": 885,
        "end_char": 902,
        "link": "https://en.wikipedia.org/wiki/Olympic_Games"
      },
      {},
      {}
    ],
    "PERSON": [],
    "TIME": []
  },
  "labels": [
    "EVENT",
    "PERSON",
    "TIME"
  ]
}
```

A WebServer segítségével megvalósított kommunikáció folyamata az alábbi szekvencia diagrammon követhető nyomon (3. ábra).



3. ábra: A modulok kommunikációja

Az ábrán csak a fontosabb HTTP metódusokat és a szükséges magyarázó függvényneveket tüntettük fel.

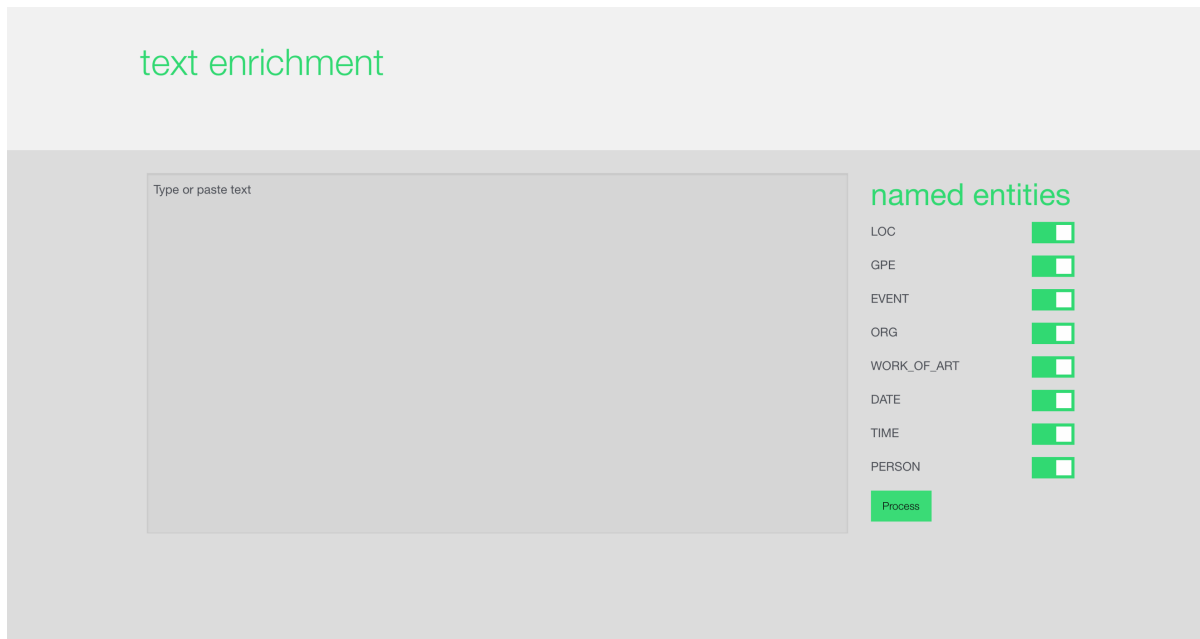
A WebServer REST API futtatásához az alábbi csomagokra van szükségünk: *requests* [9], *simplejson* [10], *flask* [11], *werkzeug* [12], *wikipedia* [15].

## User Interface

**Célja:** A felhasználó számára elérhetővé tenni az üzleti logika és a NER engine eredményeit.

A UI egyetlen komponensre a *Graphical User Interface*-re osztható. A GUI megtervezésénél nagy hangsúlyt fektettünk a letisztultságra és a harmonikus színek, formák használatára.

A UI-t a <http://127.0.0.1:5000/text-enrichment> port-on lehet elérni a WebServer futása alatt. Az adott címre történő GET metódus hatására egy HTML oldalt kapunk vissza, amely az alábbi felületet írja le (4. ábra):



4. ábra: Felhasználói felület megnyitása után

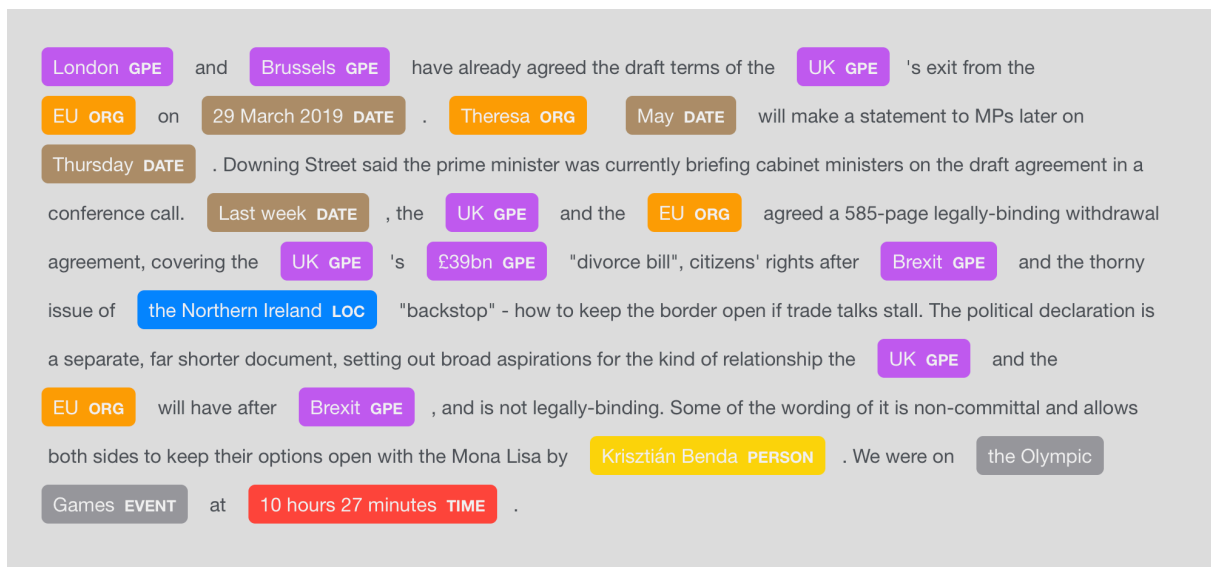
Az oldalra az névelem kategóriákat (named entities) a Jinja template engine segítségével rendereli ki a szerveren futó Flask applikáció. A feldolgozás elindulása után (Process gomb megnyomása) AJAX http POST request formájában kerül küldésre a beírt szöveg és beállított névelemek a szerver felé. Amennyiben nincs szöveg beírva vagy egyetlen elem sincs kiválasztva nem kezdődik meg a feldolgozás. A sikeres szöveg beküldést követően átjutunk az eredmények oldalra.

A GUI a WebServer klienseként működik, végig kell várnia a feldolgozás elkészültét. A felkészülés detektálására a kliens polling technikát használ, 500 ms-onként ellenőrzi, hogy előálltak-e már az adatok. Ez az idő alatt nem lehetséges semmilyen interakció. Az alábbi töltő képernyő látható (5. ábra):

# Processing...

5. ábra: Felhasználó felület a feldolgozás során

A WebServer létrehozza az objektumokat, elküldi a feldolgozandó szöveget a NER Servernek. Az elemzés és szöveggazdagítás után megkapjuk az eredményeinket. Az eredeti szövegben kiemelésre kerültek a megtalált fogalmak (6. ábra).



6. ábra: Felhasználói felület az eredményekkel

A kliens oldal egy tömbként megkapja a megtalált speciális elemeket és az eredeti szöveget. Ezekből először egy egyesített tömböt készít, amely tartalmaz egyszerű szöveg elemeket és entitásokat is. Ezen a tömbön iterálva kerülnek létrehozásra a megfelelő html elemek.

Amennyiben a kiemelt szavakra kattintunk új ablakon nyílnak meg az előállított linkek. A kiemelés színek a fogalmakhoz társított label-ekre utalnak. A kezdő felületen átállíthatók a feldolgozás során figyelembe vett label-ek (7. ábra):

## text enrichment

London and Brussels have already agreed the draft terms of the UK's exit from the EU on 29 March 2019. Theresa May will make a statement to MPs later on Thursday. Downing Street said the prime minister was currently briefing cabinet ministers on the draft agreement in a conference call. Last week, the UK and the EU agreed a 585-page legally-binding withdrawal agreement, covering the UK's £39bn "divorce bill", citizens' rights after Brexit and the thorny issue of the Northern Ireland "backstop" - how to keep the border open if trade talks stall. The political declaration is a separate, far shorter document, setting out broad aspirations for the kind of relationship the UK and the EU will have after Brexit, and is not legally-binding. Some of the wording of it is non-committal and allows both sides to keep their options open with the Mona Lisa by Krisztián Benda. We were on the Olympic Games at 10 hours 27 minutes.

### named entities

LOC	<input checked="" type="checkbox"/>
GPE	<input type="checkbox"/>
EVENT	<input type="checkbox"/>
ORG	<input type="checkbox"/>
WORK_OF_ART	<input checked="" type="checkbox"/>
DATE	<input type="checkbox"/>
TIME	<input checked="" type="checkbox"/>
PERSON	<input checked="" type="checkbox"/>

Process

7. ábra: Felhasználói felület beállítási lehetőségei

Az eredmény felületen a beállításoknak megfelelő kifejezések emelődnek ki. Az eredmények oldaláról átnavigálhatunk egy összesítő nézetre is, ahol már az egyes label-ekhez társított kifejezések láthatók csak (8. ábra):

## summary - text enrichment

### #LOC

the Northern Ireland

### #PERSON

Krisztian Benda

### #TIME

10 hours 27 minutes

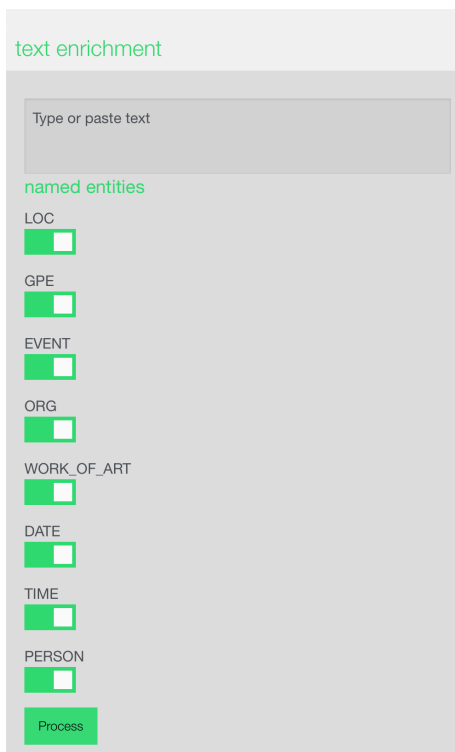
Results

New document

8. ábra: Felhasználói felület összesítő nézete

Ennek az oldalnak is a tartalma hasonlóan egy tömbben megkapott eredmények alapján kliens oldalon generálódik.

A grafikus felhasználói felület dinamikusan igazodik a más arányokkal rendelkező képernyőkhöz is (9. ábra):



9. ábra: Más képarányok támogatása

A felhasználói felület létrehozása során törekedtünk a függőségek alacsony szinten tartásához, külső könyvtárak közül a jQuery-t és a Foundation frameworkot használja csak az oldal az egyedi JavaScript és CSS mellett.

A GUI megjelenítéséhez és futtatásához egy böngészőre van szükségünk (lehetőleg Google Chrome vagy Safari, ezek alatt teszteltük).

# Telepítési leírás

A Python csomagok sajátossága, hogy sokszor külön kell telepítenünk őket. A fejlesztés során, ha felmerült egy csomag szükségessége, akkor ezeket a legtöbb esetben `pip3 install`-al telepítettük. De a helyes telepítendő parancs a célgép esetében változhat. Ezért probléma felmerülése esetén elsősorban a csomag weboldalán lehet tájékozódni.

A moduloknál szükséges csomagokat az adott modul leírásánál összeszedtük. Összegzésként itt is megtalálhatóak a használt és telepítendő csomagok:

*argparse [7], threading [8], requests [9], simplejson [10], flask [11], werkzeug [12], uuid [13], googlemaps [14], wikipedia [15], datetime [16], dateutil [17], unidecode [18], random [19], spacy [20].*

Tisztában voltunk azzal, hogy az egész rendszer telepítését jelentősen megnehezíti a különálló csomagok konfigurálása. Ezért alkalmassá tettük a rendszert docker konténerekben történő futtatásra. A folyamatot Docker telepítése után [22] a következő paranccsal lehet elindítani:

`docker-compose up`

A fenti parancs elindítja a szükséges konténereket és letölti a megfelelő függőségeket a szerverekhez. A parancs kiadása után a UI elérhető böngészőből a következő címen: <http://127.0.0.1:5000/text-enrichment>.

## A program készítése során felhasznált eszközök

- GitHub
  - Felhasználás: verziókezelésre, csapatmunka támogatására.
- Microsoft Word
  - Felhasználás: a dokumentáció elkészítésére.
- Google Chrome
  - Felhasználás: Webfejlesztés
- Microsoft Power Point
  - Felhasználás: ábra és prezentáció készítés
- Draw.io
  - Felhasználás: szekvencia és osztálydiagramm készítés
- IntelliJ IDEA:
  - Felhasználás: fejlesztőkörnyezet
- IntelliJ PyCharm:
  - Felhasználás: fejlesztőkörnyezet
- VS Code:
  - Felhasználás: fejlesztőkörnyezet

A fentiekén kívül a Telepítési leírásban említett csomagokat használtuk.



# Összefoglalás

Munkánk során megterveztük, implementáltuk illetve dokumentáltuk a Text Enrichment nevű szöveginformáció gazdagító rendszert. Az elkészített alkalmazás segítségével tetszőleges hosszúságú szövegeket dolgozhatunk fel és hozzájuk új információkat társíthatunk, ezzel segítjük az ismeretlen szövegek értelmezését és általános tájékozottság növelésére is kiválóan alkalmas.

A megvalósított alkalmazás 3 modulra bontható: User Interface, WebServer, NER Server. Az alkalmazás a User Interface-en megadott szövegeket a NER Serverben elemzi és hozzájuk névelemeket társít. A WebServer a megtalált névelemek segítségével hivatkozásokat állít össze a User Interface-en megadott szövegekhez.

Munkánk során részletes terveket készítettünk, Python és JavaScript nyelveken szerver alkalmazásokat hoztunk létre és grafikus felületet alkottunk meg. Ennek eredményeképpen egy jól működő és megbízható alkalmazást készítettünk el, amely az elvárt alapvető igényeknek megfelel, feladatát képes ellátni.

## Továbbfejlesztési lehetőségek

A Text Enrichment szoftverrel kapcsolatban számos továbbfejlesztési lehetőséget látunk. Ezek közül többet is hasznosnak tartanánk megvalósítani a tárgy keretein kívül.

Továbbfejlesztési lehetőségek a következők:

- A gyorsabb felhasználás érdekében felbontani a feldolgozást kisebb részekre és a felhasználónak a részeredményeket is elérhetővé tenni.
- Adatbázis használata, amelyben eltárolunk szöveges információkat.
- Több felhasználós működés biztosítása. A felhasználó bejelentkezés után láthatná a már feldolgozott dokumentumjait.

A fejlesztés folyamán is odafigyeltünk ezekre a továbbfejlesztési irányokra és igyekeztünk olyan tervezői döntéseket hozni, amelyek segítik a program fejlesztésének folytatását.

# Hivatkozások

- [1] SpaCy névelem felismerő rendszer: <http://spacy.io> (2018.11.24)
- [2] SpaCy modellek: <https://spacy.io/models/en> (2018.11.25)
- [3] NER Solutions Benchmarks <https://spacy.io/usage/facts-figures#benchmarks> (2018.11.24)
- [4] SpaCy en\_core\_web\_sm modell: [https://spacy.io/models/en#en\\_core\\_web\\_sm](https://spacy.io/models/en#en_core_web_sm) (2018.11.25)
- [5] SpaCy en\_core\_web\_md modell: [https://spacy.io/models/en#en\\_core\\_web\\_md](https://spacy.io/models/en#en_core_web_md) (2018.11.25)
- [6] SpaCy en\_core\_web\_lg modell: [https://spacy.io/models/en#en\\_core\\_web\\_lg](https://spacy.io/models/en#en_core_web_lg) (2018.11.25)
- [7] Python argparse Library: <https://pypi.org/project/argparse/> (2018.11.25)
- [8] Python threaded Library: <https://pypi.org/project/threaded/> (2018.11.25)
- [9] Python Requests: <http://docs.python-requests.org/en/master/user/install/> (2018.11.25)
- [10] Python simplejson: <https://pypi.org/project/simplejson/> (2018.11.25)
- [11] Python Flask: <http://flask.pocoo.org/docs/1.0/installation/> (2018.11.25)
- [12] Python Werkzeug: <https://pypi.org/project/Werkzeug/> (2018.11.25)
- [13] Python uuid: <https://pypi.org/project/uuid/> (2018.11.25)
- [14] Python googlemaps: <https://pypi.org/project/googlemaps/> (2018.11.25)
- [15] Python wikipedia: <https://pypi.org/project/wikipedia/> (2018.11.25)
- [16] Python datetime: <https://pypi.org/project/DateTime/> (2018.11.25)
- [17] Python datutil: <https://pypi.org/project/python-dateutil/> (2018.11.25)
- [18] Python unicode: <https://pypi.org/project/Unidecode/> (2018.11.25)
- [19] Python random: <https://pypi.org/project/random2/> (2018.11.25)
- [20] Python SpaCy installation: <https://spacy.io/usage/> (2018.11.25)
- [21] ZURB Foundation 6: <https://foundation.zurb.com> (2018.11.25)
- [22] Docker instalálás: <https://docs.docker.com/install/> (2018.11.25)