Recruitment Task

#recruitment

1. Извличане на информация от неструктуриран текст

Във файла `a_portrait_new.txt` се съдържа текста на романа *"A portrait of the painter as a yong man"*. Задачата е да се извлече програматично информация за имената на героите в нея и техните споменавания.

- [ ] Отвори текстовия файл и извлечи съдържанието му в подходящ формат, разделяйки го на изречения.
- [ ] Валидирай UTF-8 енкодинга и изчисти/замести всякакви невалидни символи в текста.
- [ ] Извади всички именувани обекти в текста, открити от **готов** (и по възможност лек) Named Entity Recognition (NER) модел - например от Spacy. Този туториал е достатъчен за да се ориентираш в употребата му за целта на задачата.
- [ ] Филтрирай само имената на хора (*герои*) извадени с NER модела, идентифицирай и стандартизирай всякакви частични съкратени споменавания към пълната форма в която се срещат.
- [ ] В колко и кои изречения се с среща всяко едно име на герой?
- [ ] Заедно с кои други герои се споменава в едно и също изречение всеки даден герой и колко пъти?
- [ ] Съхрани в подходяща структура от данни информация за споменаванията на героите в изреченията от романа.
- [ ] Използвайки тази структура, можеш ли да откроиш даден герой като главен, второстепенен или епизодичен на база на извлечената информация?
- [ ] **ЗА БОНУС ТОЧКИ:** Приложи техника за сравнение и групиране на изреченията в текста по прилика - базирана на количествата споменати в тях думи върху лексикалния речник на книгата, и/или на други мярка (изненадай ни). Използвай я за да изследваш различни *типове* изречения в които се споменават героите. Импровизирай с прилагането на това типизиране в досега изведения принцип за категоризация на ранга на героите.

2. Доставяне на услугата през API

Създай REST уеб приложение с GET API използвайки WSGI/Flask/Gunicorn със следните endpoints за доставка на услуги:

- [ ] `getCharacterInfo` - очаква `name` параметър и в случай, че той съвпада с име на герой от книгата, връща JSON схема съдържаща:

- `rank` : неговия ранг (1-3);
- `mentions_count` : броя на всички негови споменавания;
- `mono_mentions_count` : броя изречения, в които името на героя се среща само;
- `co_mentions_count` : броя изречения, в които то се среща заедно с други имена на герои;
- `co_mention_names` : имената на героите заедно с които се среща в едно и също изречение;

☐ `getMainCharacters` - връща JSON схемата от `getCharacterInfo` за всички главни (ранг-1) герои;

☐ `getSupportCharacters` - връща JSON схемата от `getCharacterInfo` за топ-10 второстепенни (ранг-2) герои;

☐ `getEpisodeCharacters` - връща JSON схемата от `getCharacterInfo` за 10 произволни епизодични (ранг-3) герои;

☐ `getCharacterMenitons` - очаква `name` параметър и в случай, че той съвпада с име на герой от книгата, връща списък с всички изречения, в които името му се споменава; в случай че не получи `name` параметър връща такива списъци за всички главни герой;

☐ `getCharactersCoMenitons` - очаква `name_a`, `name_b` параметри и в случай, че те съвпадат с имена на герои от книгата, връща всички изречения, в които те се споменават заедно, ако има такива.

☐ **ЗА БОНУС ТОЧКИ** добави endpoint с нова API функционалност по своя идея.

Очаква се всеки API endpoint на приложението да връща валиден отговор съдържащ съобщение/информация за грешка в случай, че получи невалидна заявка или стойност на параметър, която не съвпада с име на герой от книгата.

Изисквания

- Използвай Python $\geq$ 3.6 и всякакви съвместими open source пакети които желаеш.
- Кода и приложението трябва да могат да се изпълнят на стандартен лаптоп.
- Можеш да използваш Jupyter notebooks за бърза прототипизация и/или достъпна публикация на решението на задачите от част 1, но това не е задължително и не е критерий за оценка на решението.
- Кодът за решението и на двете части трябва да бъде качен заедно с всякакви съпътстващи файлове в github.

- Задължително изготви `README.md` документация, описваща изпълнението на кода от част 1, пускането и употребата на приложението от част 2.
- *Всякаква друга написана или автоматично генерирана endpoint или functional документация, добре форматиран output или визуализация биха били в плюс!*
- *Няма изискване за съхраняване на извлечената информация от част 1 в какъвто и да е вид база данни за целите на част 2, но имаш свободата да използваш всякакъв удобен за теб модел или да демонстрираш опита си в тази област ако желаеш.*
- ***TIP:** Решенията на задачите от част 2 стъпват върху тези от част 1, но тестват и оценяват различни умения и е добре в случай, че не си сигурен в резултатите си от част 1, независимо от тях да предоставиш възможно най-добре работещо приложение в част 2!*
- ***Забележка:** Решения, които не могат да бъдат изпълнени, няма да бъдат разглеждани!*