6. domácí úkol

- zadaný: 4. 12. 2023
- deadline: 18. 12. 2023, **23:55**
- maximální počet bodů: 40 (za funkcionalitu, EduLint)

Pokyny

- Odevzdejte jediný soubor pojmenovaný prijmeni_du6.py (“prijmeni” nahraďte svým příjmením) s definicemi požadovaných funkcí. Odevzdávárnu najdete v ISu: Student / FI:IB113 / Odevzdávárny / Skupina 10 / DU6
- Šablona řešení je dostupná ve studijních materiálech.
- Jména odevzdávaných funkcí neměňte; ať zůstanou pojmenované stejně, jako jsou pojmenované v zadání.
- Případné dotazy pokládejte v diskuzním fóru této skupiny ve vláknu k tomuto úkolu.
- Příklad vypracujte samostatně. Nepovolená spolupráce bude potrestána dle pravidel předmětu.
- Funkce, které napíšete, můžete používat v rámci jiných funkcí. Můžete si implementovat i další pomocné funkce.
- Pokud chcete své řešení předvést na konkrétních textech/obrázcích, odevzdejte je spolu s řešením.

Zadání

Zpracování dat

V tomto úkolu provedete analýzu n-gramů v textu (n-gram je posloupnost n po sobě jdoucích slov).

Vaším úkolem bude definovat funkci `frequent_ngrams(path, n, k, stopwords)`. Tato funkce přečte obsah souboru v souboru s cestou path. Každý řádek souboru rozdělí podle mezer a pro dál zpracovává jednotlivá slova. Z každého slova nejdřív odebere interpunkci (pro naše potřeby za interpunkci považujeme znaky . (tečka), , (čárka), ? (otazník), ! (vykřičník), " (rovné uvozovky dvojité) a ' (rovné uvozovky jednoduché). Z takto očištěného proudu slov pak budete skládat n-gramy.

Z řádku “My hovercraft is full of eels.” po očištění vznikne sekvence “My”, “hovercraft”, “is”, “full”, “of”, “eels” (žádná tečka za eels). Tato sekvence obsahuje pět 2-gramů (My, hovercraft), (hovercraft, is), (is, full), (full, of) a (of, eels), nebo čtyři 3-gramy (My, hovercraft, is), (hovercraft, is, full), (is, full, of), (full, of, eels), atd.

Jak velké n-gramy máte uvažovat udává parametr `n`.

Funkce má pro zadaný vstup vrátit `k` nejčastějších `n`-gramů a jejich četností ze souboru s cestou `path`, které neobsahují žádné slovo z množiny `stopwords`.

Při počítání nezohledňujte, jestli je slovo napsané s velkým nebo malým písmenem.

Pokud má více n-gramů stejnou četnost, preferujte ty dříve v abecedě. N-gramy se řadí podle slov, která obsahují: pokud je první slovo v n-gramu menší než první slovo v druhém ngramu, pak je celý n-gram menší. Pokud jsou stejné, rozhoduje se podle další složky, atd. Slova se řadí tak, že pokud dva řetězce začínají na různá písmena, první je slovo s písmenem dříve v abecedě, pokud na stejné písmeno, seřaďte je podle dalšího; pokud jeden z řetězců už další znak nemá, pak by ten kratší měl být dřív.
(Tato pravidla jsou konzistentní s tím, jak Python řadí ntice a řetězce defaultně, abyste je dodrželi, stačí použít vestavěnou funkci `sorted` a její parametr `key`).

Uvažujme soubor s názvem `duck.txt`, který obsahuje jediný řádek s textem:
`He is lord, the duck is lord, he is lord, lord, lord, the duck is lord.`

Volání `frequent_ngrams("duck.txt", 3, 10, set())` vrátí
```
[
  (("duck", "is", "lord"), 2),
  (("he", "is", "lord"), 2),
  (("lord", "the", "duck"), 2),
  (("the", "duck", "is"), 2),
  (("is", "lord", "he"), 1),
  (("is", "lord", "lord"), 1),
  (("is", "lord", "the"), 1),
  (("lord", "he", "is"), 1),
  (("lord", "lord", "lord"), 1),
  (("lord", "lord", "the"), 1)
]
```
Volání `frequent_ngrams("duck.txt", 3, 10, {"is"})` vrátí
```
[
  (("lord", "the", "duck"), 2),
  (("lord", "lord", "lord"), 1),
  (("lord", "lord", "the"), 1)
]
```
Volání `frequent_ngrams("duck.txt", 2, 3, set())` vrátí
```
[
    (("is", "lord"), 4),
    (("duck", "is"), 2),
    (("he", "is"), 2)
]
```
Volání `frequent_ngrams("duck.txt", 2, 3, {"the", "is"})` vrátí
```
[
    (("lord", "lord"), 2),
    (("lord", "he"), 1)
]
```

Pokud vaše řešení nebude splňovat všechny požadavky (například nebude řadit podle slov, jen podle četností, nebude filtrovat stopwords, bude rozlišovat velikosti písmen, apod.), stále dostanete za příklad část bodů. V takovém případě ale v řešení uveďte, co nefunguje.

Regulární výrazy

Vyřešte 15 příkladů na https://www.umimeinformatiku.cz/regularni-vyrazy. Alespoň pět z nich musí být ze sady Mix: střední. Tato řešení neodevzdávejte jako součást odevzdávaného souboru.

Bitmapový obrázek: detekce hran

Napište funkci `detect_edges`, která pro zadaný obrázek vygeneruje obrázek znázorňující hrany (ostré přechody) ve vstupním obrázku (vizte ukázku). Funkce jako první argument bere cestu k obrázku, zbývající parametry jsou na vás.

Detekce hran je v obecnosti celkem obtížný problém v počítačové grafice, nemusíte si však studovat žádné složité algoritmy. Stačí, když naprogramujete přímočarou variantu ve smyslu "hrana ⇔ bod má výrazně odlišnou barvu od bezprostředně okolních bodů". Konkrétní postup a hranici, co znamená “výrazně odlišnou”, implementujte podle vlastního uvážení.

Nebojte se odevzdat řešení, které podle vás nejlépe detekuje hrany, ne nutně takové, které je nejpodobnější příkladu níže.