

به نام خدا

نام : حسین سیم چی

شماره ی تمرین : سوم

نام درس : پردازش زبان های طبیعی

تاریخ ارسال : ۱۳۹۸/۱۰/۱

مقدمه :

در مورد تمرین و شرح کارهای انجام شده در ابتدا باید این نکته را متذکر شوم که تمرین انجام شده با استفاده از ابزارهای پردازش زبان **هضم** و **Stanfordnlp** صورت گرفته است ، به این صورت که در ابتدا برای انجام دادن ابزار **Stanfordnlp**، متن داده شده را تمیز و نرمال کرده و سپس مراحل نحوی آن که شامل POS TAG میباشد را انجام میدهیم . همچنین همین کار برای استفاده ابزار **هضم** نیز صورت گرفته است .

مراحل انجام کار :

در ابتدا باید با توجه به اینکه متن ما تمیز و قابل پردازش نمیباشد باید آن را با استفاده از کد نوشته

شده تمیز کرد و سپس آن را برای پردازش آماده نمود.

باتوجه به عکس زیر که از کد گرفته شده است خواهیم داشت :

The screenshot displays the PyCharm IDE interface. The top toolbar includes standard editing and development tools like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. Below the toolbar, the left sidebar shows the Project view with a tree structure containing folders like jetbrains, venv, and files such as Can't sleep, Gonna die_noise_10.jpg, fddd.py, g.py, grayimage.jpg, jo.h.py, m.py, External Libraries, and Scratches and Consoles.

The main editor window shows the code for fddd.py. The code defines a function replace(self, word) that uses wordnet.synsets(word) to find synonyms and returns the first one if found. It also implements a Replacer class with methods replace(y) and repeat_regexp.sub(self.repl, word). The main logic involves processing a string y by removing unwanted characters (alpha, digit, punctuation) and then printing the result.

```
def replace(self, word):
    if wordnet.synsets(word):
        return word
    repl_word = self.repeat_regexp.sub(self.repl, word)
    if repl_word != word:
        return self.replace(repl_word)
    else:
        return repl_word

replacer = Replacer()
replacer.replace(y)
unwanted_alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
for alpha in unwanted_alpha:
    y = y.replace(alpha, "")
unwanted_digit = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
for digit in unwanted_digit:
    y = y.replace(digit, "")
unwanted_punc = ["'", '"', '=', '@', '&', '%', '^', '~', '\\", '$', '!', '<', '>', '?', '{', '}', ':', ';', '\\n', '\\t', '(', ')', '[', ']', ',', '.', '-', '+', '#', '\\u200c', '\\ufe0f']
for punc in unwanted_punc:
    y = y.replace(punc, "")

# To print size after Remove extra characters
print("print size after Remove extra characters is {}".format(len(y)))
```

The bottom panel shows the Run console output for the command "C:\Users\Lenovo\AppData\Local\Programs\Python\Python37\python.exe "G:/PyCharm 2019.2.3/fddd.py"". The output indicates the original string size is 31181, the size after removing extra characters is 29740, and it proceeds with tokenization and model loading settings.

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python37\python.exe "G:/PyCharm 2019.2.3/fddd.py"
Hello , my name is Hossein Simchi
print original size : 31181
print size after Remove extra characters is 29740
Use device: cpu
---
Loading: tokenize
With settings:
{'model_path': 'C:\\Data\\fa_seraji_models\\fa_seraji_tokenizer.pt', 'lang': 'fa', 'shorthand': 'fa_seraji', 'mode': 'predict'}
```

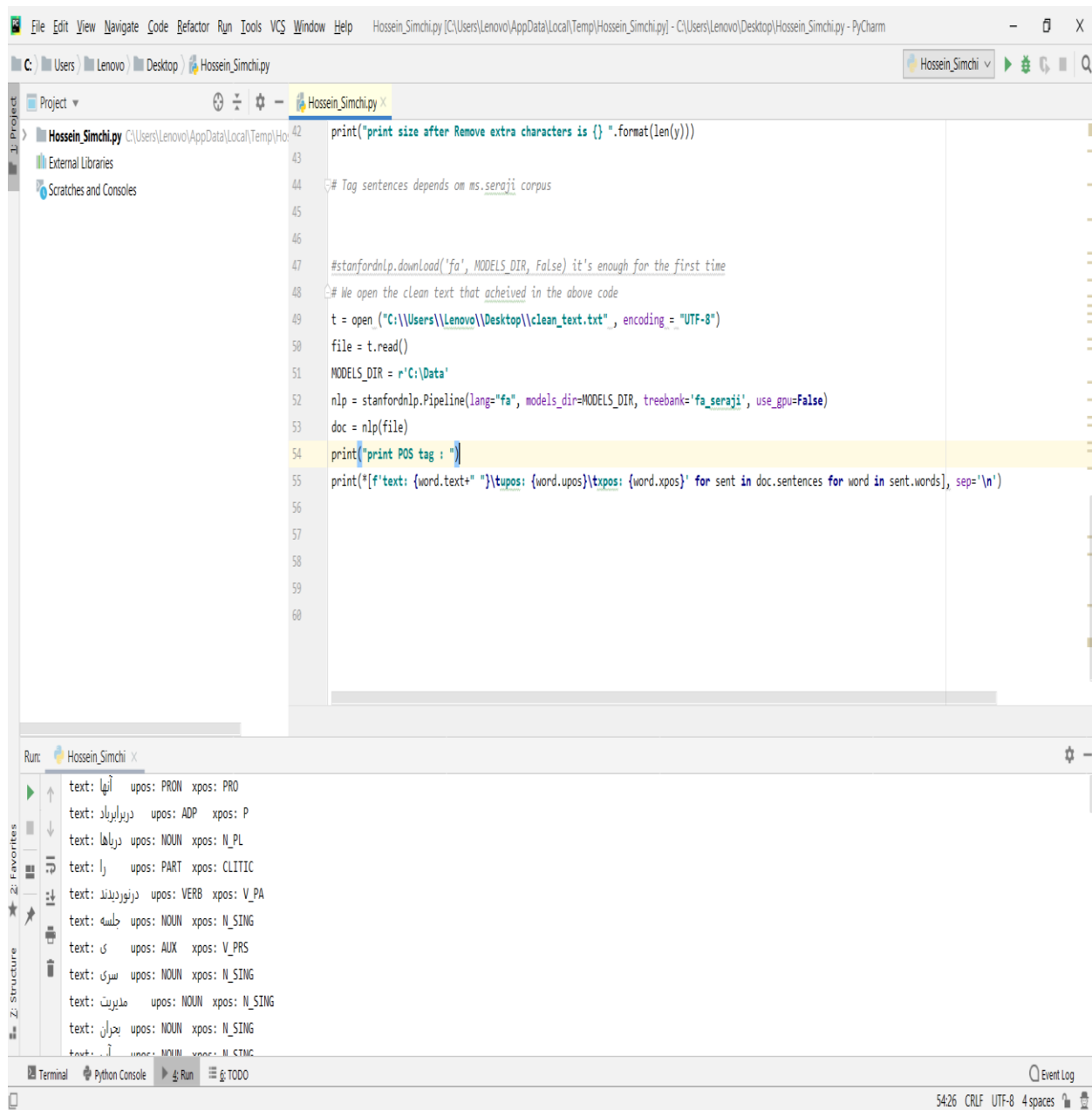
همانطور که از کد بالا معلوم است در ابتدا فایل را خوانده و سپس با استفاده از کد نوشته شده آن را تمیز می کنیم

اگر طول متن اولیه را محاسبه کنیم ، پس از فرآیند تمیز کردن آن خواهیم دید که طول آن کاهش یافته است یعنی پس از انجام فرایند تمیز کردن خواهیم داشت :

```
Hello , my name is Hossein Simchi  
print original size : 31181  
print size after Remove extra characters is 29740
```

تصاویر گرفته شده تماما از متن کد میباشد که در سورس اصلی برنامه که ارسال شده است نیز قابل مشاهده میباشد .

پس از آنکه متن کاملاً تمییز و نرمال شد حالا میتوان باتوجه به **پیکره ی خانم دکتر سراجی** که استفاده از آن برای ابزار **StanfordNlp** میسر میباشد پیکره ی خود را تگ زد :



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Hossein_Simchi.py [C:\Users\Lenovo\AppData\Local\Temp\Hossein_Simchi.py] - C:\Users\Lenovo\Desktop\Hossein_Simchi.py - PyCharm
C:\Users\Lenovo\Desktop\Hossein_Simchi.py
Project
Hossein_Simchi.py
External Libraries
Scratches and Consoles
42 print("print size after Remove extra characters is {}".format(len(y)))
43
44 # Tag sentences depends on ms.seraji corpus
45
46
47 #stanfordnlp.download('fa', MODELS_DIR, False) it's enough for the first time
48 # We open the clean text that achieved in the above code
49 t = open("C:\\Users\\Lenovo\\Desktop\\clean_text.txt", encoding = "UTF-8")
50 file = t.read()
51 MODELS_DIR = r"C:\Data"
52 nlp = stanfordnlp.Pipeline(lang="fa", models_dir=MODELS_DIR, treebank='fa_seraji', use_gpu=False)
53 doc = nlp(file)
54 print("print POS tag : ")
55 print([f'text: {word.text+" "}{tupos: {word.upos}}{txpos: {word.xpos}}' for sent in doc.sentences for word in sent.words], sep='\n')
56
57
58
59
60
Run: Hossein_Simchi
text: آنها upos: PRON xpos: PRO
text: دریافت upos: ADP xpos: P
text: دریاها upos: NOUN xpos: N_PL
text: را upos: PART xpos: CLITIC
text: درنوردیدند upos: VERB xpos: V_PA
text: جلسه upos: NOUN xpos: N_SING
text: ی upos: AUX xpos: V_PRS
text: سری upos: NOUN xpos: N_SING
text: مدیریت upos: NOUN xpos: N_SING
text: چران upos: NOUN xpos: N_SING
text: آن upos: NOUN xpos: N_SING
Terminal Python Console Run TODO Event Log
5426 CRLF UTF-8 4 spaces
```

پیکره تگ خورده به صورت فایل Text ارسال میگردد

هضم:

در گام بعدی با استفاده از ابزار **هضم** پیکره ی خود را ابتدا تمیز نموده و سپس بخش **تحلیل صرفی** ،
سطحی و نحوی آن را انجام میدهیم . باتوجه به اینکه برای بخش تحلیل نحوی پیکره ی ما بیش از
حد بزرگ است ۴۰ جمله رندم از آن را انتخاب کرده و تحلیل نحوی آنها را انجام میدهیم .
برای بخشهای دیگر ذکر شده کل متن را تحلیل نموده ام که فایل های آن به صورت Text ارسال
میگردد .

برای نمونه ی کار بخش تحلیل نحوی داریم :



مقایسه دقت و خطای دو ابزار ذکر شده :

در ابزار **Stanfordnlp** همانطور که بیان شد ، تگ هر کلمه را براساس پیکره ی خانم دکتر سراجی و براساس **UnigramTagger** انجام میدهد به این صورت که برای هر کلمه ، پیکره ی خانم دکتر سراجی را نگاه کرده و باتوجه به بیشترین احتمال تگ برای آن کلمه ، تگ را به کلمه ی موردنظر نسبت میدهد . دقت اینکار با استفاده از مدل بیان شده حدود **۸۵** درصد میباشد حال اگر متن را باتوجه به مدل های **TrigramTagger** و **BigramTagger** تگ بزنیم خواهیم دید که دقت انجام کار **کاهش** می یابد . بهترین کار برای **افزایش دقت ترکیب این مدلها** باهم میباشد که پس از انجام این کار مشاهده خواهیم نمود که دقت **نزدیک ۹۰** درصد خواهد رسید .

اگر متن را به این صورت بدهیم که بعضی از کلمات به فعل ها و کلمات دیگر متصل باشد خواهیم دید که ابزار **Stanfordnlp نمیتواند** به درستی تگ را اختصاص دهد و باتوجه به اینکه دو کلمه ی متصل شده را در پیکره مرجع پیدا نمیکند ، تگ آن را **ناشناس** یا در بعضی موارد همانطور که قابل مشاهده است **Noun** تشخیص میدهد ولی در ابزار **هضم** کمی بهتر عمل میکند و در بخش تجزیه نحوی به طور بهتری اجزای جملات را از هم جدا میکند

در کل باتوجه به ارزیابی های صورت گرفته بنظر ابزار هضم دقت بالاتر و بهتری را ارائه میدهد هرچند همانطور که گفته شد اگر از ترکیب مدل های نام برده شده استفاده کنیم ، ابزار

Stanfordnlp دقتی **حدود ۹۰** درصد را برمیگرداند که نسبتا دقت خوب و قابل قبولی است .

*****توجه :** تمامی فایل ها اعم از متن اصلی (**dataset**) ، متن تمیز شده (**clean_text**) و سورس

کدهای نوشته شده به صورت جدا ارسال میگردد . همچنین خروجی کدهای نوشته شده نیز به

صورت جدا ارسال میگردد