دانشکده مهندسی کامپیوتر

پردازش زبان‌های طبیعی

تمرین دوم

دکتر احسان‌الدین عسگری

گروه ۱، آرین احدی نیا، محمدمهدی ابوترابی، ارشان دلیلی

۱۲ آذر ۱۴۰۱

فهرست مطالب

در این تمرین، ابزاری جهت استخراج اطلاعات رزومه با استفاده از عبارات منظم برای زبان فارسی توسعه می‌دهیم.

۱ پیش‌پردازش

برای پیش‌پردازش از ابزار نرمالیزیشن کتاب‌خانه hazm استفاده می‌کنیم. البته در بعضی از ماژول‌ها و تسک‌ها پیش‌پردازش‌های مخصوص آن را اعمال می‌کنیم.

۲ استخراج نام

برای استخراج نام، ماژول NameDetector را می‌سازیم. این ماژول با دریافت اسامی کراول شده در اینترنت (شامل نام و نام‌خانوادگی) و پترن‌های مشخص نام، شروع به استخراج نام‌ها از رزومه می‌کند. تصویر این ماژول را در زیر مشاهده می‌نمایید. این ماژول توانایی پیدا کردن نام‌های ترکیبی و نام‌هایی که نام خانوادگی آن در دیتاست نام‌های خانوادگی وجود ندارد را نیز دارا می‌باشد. (لازم به ذکر است دیتاست نام در فولدر resources قرار دارد.)

```python
import json
import re


class NameDetection:
    def __init__(self):
        with open('cv_info_extractor/resources/last_names_regex.json', 'r', encoding="utf-8") as file:
            self.last_names_reg = json.loads(file.read())

        with open('cv_info_extractor/resources/first_names_regex.json', 'r', encoding="utf-8") as file:
            self.first_names_reg = json.loads(file.read())

        self.pattern = f"(^|\W)((((({self.first_names_reg})(\W+))+)({self.last_names_reg}))|((((({self.first_names_reg})(\W+))+)(\w+))|({self.last_names_reg}))|({se

    def match_name(self, inp):
        matches = []
        count_pattern = self.pattern.format()
        for matched in re.finditer(count_pattern, inp):
            start, end = matched.span()
            inp = inp[:start] + '#' * (end - start) + inp[end:]
            matches.append(matched)
        return matches

    def find_name(self, text):
        matched_names = self.match_name(text)
        if not matched_names:
            return ['Not Found'] * 3
        full_name = matched_names[0].group().strip()
        if matched_names[0].groups()[3]:
            first_name = matched_names[0].groups()[3].strip()
        elif len(matched_names[0].groups()) > 9 and matched_names[0].groups()[9]:
            first_name = matched_names[0].groups()[9].strip()
        else:
            first_name = 'None'
        if len(matched_names[0].groups()) > 7 and matched_names[0].groups()[7]:
            last_name = matched_names[0].groups()[7].strip()
        elif matched_names[0].groups()[-4]:
            last_name = matched_names[0].groups()[-4].strip()
        else:
            last_name = 'None'
        return full_name, first_name, last_name
```

شکل ۱: تصویر ماژول نام

۳ استخراج ایمیل

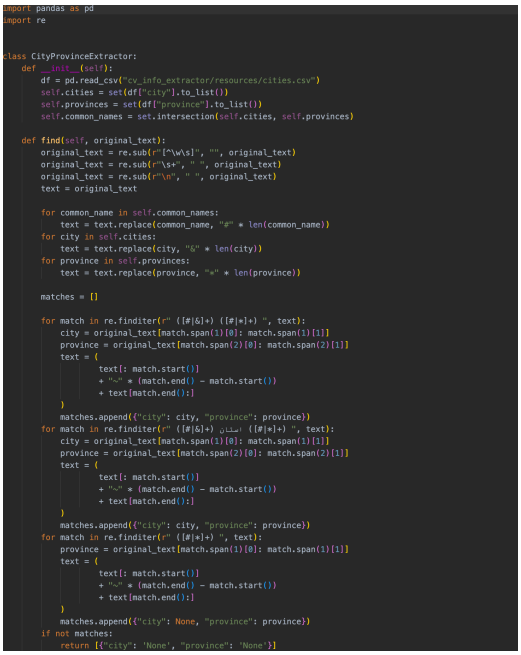
برای استخراج ایمیل، ماژول EmailDetector را می‌سازیم. این ماژول با پترن‌های ایمیل شروع به استخراج ایمیل از رزومه می‌کند. تصویر این ماژول را در زیر مشاهده می‌نمایید.

```python
class EmailDetection:
    def __init__(self):
        self.pattern = r"\b(\w+([-+(\.|\[dot\])']\w+)*(@|\[at\])\w+([-(\.|\[dot\])]\w+)*(\.|\[dot\])\w+([-(\.|\[dot\])]\

    def match_email(self, inp):
        matches = []
        count_pattern = self.pattern.format()
        for matched in re.finditer(count_pattern, inp):
            start, end = matched.span()
            inp = inp[:start] + '#' * (end - start) + inp[end:]
            matches.append(matched)
        return matches

    def find_email(self, text):
        matched_emails = self.match_email(text)
        if not matched_emails:
            return 'Not Found'
        return matched_emails[0].group().strip()
```

شکل ۲: تصویر ماژول ایمیل

۴ استخراج شهر

برای استخراج شهر، ابتدا اسامی را که بین شهرها و استان‌ها مشترک هستند (مانند تهران و یزد) کراول می‌کنیم و در فایل cities.csv ذخیره می‌کنیم. سپس کل متن را پیمایش می‌کنیم و اسامی شهرها، استان‌ها و اسامی مشترک را mask می‌کنیم. سپس با استفاده از الگوهای این ماژول، نام شهرها و استان‌ها را استحراج می‌کنیم.



```python
import pandas as pd
import re


class CityProvinceExtractor:
    def __init__(self):
        df = pd.read_csv("cv_info_extractor/resources/cities.csv")
        self.cities = set(df["city"].to_list())
        self.provinces = set(df["province"].to_list())
        self.common_names = set.intersection(self.cities, self.provinces)

    def find(self, original_text):
        original_text = re.sub(r"[^\w\s]", "", original_text)
        original_text = re.sub(r"\s+", " ", original_text)
        original_text = re.sub(r"\n", " ", original_text)
        text = original_text

        for common_name in self.common_names:
            text = text.replace(common_name, "#" * len(common_name))
        for city in self.cities:
            text = text.replace(city, "&" * len(city))
        for province in self.provinces:
            text = text.replace(province, "*" * len(province))

        matches = []

        for match in re.finditer(r" ([#|&]+) ([#|*]+) ", text):
            city = original_text[match.span(1)[0]: match.span(1)[1]]
            province = original_text[match.span(2)[0]: match.span(2)[1]]
            text = (
                    text[: match.start()]
                    + "^" * (match.end() - match.start())
                    + text[match.end():]
            )
            matches.append({"city": city, "province": province})
        for match in re.finditer(r" ([#|&]+) استان ([#|*]+) ", text):
            city = original_text[match.span(1)[0]: match.span(1)[1]]
            province = original_text[match.span(2)[0]: match.span(2)[1]]
            text = (
                    text[: match.start()]
                    + "^" * (match.end() - match.start())
                    + text[match.end():]
            )
            matches.append({"city": city, "province": province})
        for match in re.finditer(r" ([#|*]+) ", text):
            province = original_text[match.span(1)[0]: match.span(1)[1]]
            text = (
                    text[: match.start()]
                    + "^" * (match.end() - match.start())
                    + text[match.end():]
            )
            matches.append({"city": None, "province": province})
        if not matches:
            return [{"city": 'None', "province": 'None'}]
```

شکل ۳: تصویر ماژول استخراج شهر

۵ استخراج تاریخ تولد

برای استخراج تاریخ تولد، ابتدا با استفاده از الگوهای تاریخ شروع به استخرا تاریخ می‌کنیم. سپس با چک کردن مکان کلمات مرتبط با تاریخ تولد، نزدیک‌ترین تاریخ (که تا ۱۰۰ کاراکتر بعد می‌تواند باشد) را برمی‌گردانیم.

```python
import re
from persiantools import digits


class DateDetection:
    def __init__(self):
        self.pattern1 = r'\d{1,2}(\s*)?(?:فروردین|اردیبهشت|خرداد|تیر|مرداد|شهریور|مهر|آبان|آذر|دی|بهمن|اسفند)(\s*)?\d{2,4}'
        self.pattern2 = r'\d{2,4}(\s*)?(?:فروردین|اردیبهشت|خرداد|تیر|مرداد|شهریور|مهر|آبان|آذر|دی|بهمن|اسفند)(\s*)?\d{1,2}'
        self.pattern3 = r'\d{2,4}/\d{1,2}/\d{1,2}'
        self.pattern = f'{self.pattern1}|{self.pattern2}|{self.pattern3}'
        self.keywords = [
            'متولد',
            'تولد',
            'زادروز',
        ]

    def match_date(self, inp1):
        matches = []
        inp = digits.fa_to_en(inp1)
        for matched in re.finditer(self.pattern, inp):
            start, end = matched.span()
            inp = inp[:start] + '#' * (end - start) + inp[end:]
            matches.append(matched.group().strip())
        return matches

    def find_date_number(self, text):
        for keyword in self.keywords:
            if keyword not in text:
                continue
            text1 = text[text.find(keyword):]
            matched_dates = self.match_date(text1)
            if not matched_dates:
                matched_dates = self.match_date(text1[::-1])
                if not matched_dates:
                    return 'Not Found'
                return matched_dates[0]
            return matched_dates[0]
        return 'Not Found'
```

شکل ۴: تصویر ماژول استخراج تاریخ تولد

۶ استخراج شماره تلفن

برای استخراج شماره تلفن، ابتدا الگوهای متداول شماره تلفن و انواع مرسوم نوشتن شماره را به دست آورده و با استفاده از این الگوها، شماره تلفن را به دست می‌آوریم.

```python
import re
from persiantools import digits


class PhoneNumberDetection:
    def __init__(self):
        self.pattern1 = r"(^|\W)((\+98|0)?9\d{9})($|\W)"
        self.pattern2 = r"(^|\W)((\+98|0)(\s)?9\d{9})($|\W)"
        self.pattern3 = r"(^|\W)((\+98|0)(\s)?(9\d{2})(\s)?(\d{4})(\s)?(\d{3}))($|\W)"
        self.pattern4 = r"(^|\W)(\(\+98\)?9\d{9})($|\W)"
        self.pattern5 = r"(^|\W)((\+98|0)(\s)*(9\d{2})(\s)*(\d{3})(\s)*(\d{4}))($|\W)"
        self.pattern = f'{self.pattern1}|{self.pattern2}|{self.pattern3}|{self.pattern4}|{self.pattern5}'

    def match_phone_number(self, inp1):
        matches = []
        inp = digits.fa_to_en(inp1)
        for matched in re.finditer(self.pattern, inp):
            start, end = matched.span()
            inp = inp[:start] + '#' * (end - start) + inp[end:]
            matches.append(matched)
        return matches

    def find_phone_number(self, text):
        matched_phones = self.match_phone_number(text)
        if not matched_phones:
            matched_phones = self.match_phone_number(text[::-1])
            if not matched_phones:
                return 'Not Found'
            return matched_phones[0].group().strip()
        return matched_phones[0].group().strip()
```

شکل ۵: تصویر ماژول استخراج شماره تلفن

۷ استخراج وضعیت شغلی

برای استخراج وضعیت شغلی، از تعدادی کلمات کلیدی استفاده می‌کنیم. سپس با استفاده از تعدادی حالت پیش‌فرض برای وضعیت شغلی، در صورت وجود آن‌ها، برگردانده می‌شوند و در غیر این‌صورت، در سوابق شغلی شروع به بررسی تعدادی کلمات کلیدی با مضمون "تا کنون" می‌کند و در صورت وجود آن‌ها را برمی‌گرداند.

هم‌چنین برای حقوق مورد انتظار و نوع شغل مورد نظر نیز ماژول‌های جداگانه ساخته شده که از روی کلمات کلیدی و مقادیر مشخص اقدام به استخراج این موارد می‌نماید که ساختاری بسیار شبیه به وضعیت اشتغال دارد.

```python
class EmploymentStatusExtractor:
    def __init__(self) -> None:
        keys = [
            "وضعیت اشتغال",
            "وضعیت اشتغالی",
            "وضعیت استخدام",
            "وضعیت استخدامی",
            "وضعیت کار",
            "وضعیت کاری",
        ]
        values = [
            "علاقه مند",
            "علاقه‌مند",
            "علاقه مند",
            "بیکار",
            "مشغول",
            "شاغل",
        ]
        up_to_now = [
            "تاکنون",
            "تا اکنون",
            "تا زمان حال",
            "تا حال",
            "تا الان",
            "تا الآن",
        ]
        self.pattern_keys = r"({})\s*:*".format("|".join(keys))
        self.pattern_values = r"({})".format("|".join(values))
        self.pattern_up_to_now = r"({})".format("|".join(up_to_now))

    def find(self, text1, job_text):
```

شکل ۶: تصویر ماژول وضعیت شغلی

۸ استخراج عنوان شغلی

در این بخش ما دیتاستی از عناوین شغلی مختلف که در رزومه‌های افراد پرتکرار است جمع‌آوری کردیم. سپس این عناوین شغلی را با انجام پیش‌پردازش مناسب با بخش سرآیند و بخش سوابق شغلی رزومه تطبیق می‌دهیم و اولین مورد انطباق را به عنوان خروجی برمیگردانیم. این کار از این جهت منطقی است که سوابق شغلی به ترتیب نزولی بر حسب تاریخ و اهمیت مرتب می‌شود بنابرین اولین مورد انطباق، آخرین سابقه شغلی فرد است.

```python
import pandas as pd


class JobTitleFinder:
    def __init__(self) -> None:
        titles = list(
            pd.read_csv(
                "cv_info_extractor/resources/jobs.csv",
            )["title"]
        )
        expanded_1 = [
            text.replace("مهندس", "مهندسی") for text in titles if "مهندس" in text
        ]
        titles += expanded_1
        self.titles = sorted(titles, key=lambda s: -len(s))
        self.normalized_title = [self.normalize(text) for text in self.titles]

    def normalize(self, txt):
        txt = txt.replace(" ", "")
        txt = txt.replace("\n", "")
        txt = txt.replace("\u200C", "")
        return txt

    def find(self, text):
        normalized_text = self.normalize(text)
        min_index = float("inf")
        for idx, title in enumerate(self.normalized_title):
            if title in normalized_text:
                index = normalized_text.index(title)
                if index < min_index:
                    return self.titles[idx]
        return 'None'
```

شکل ۷: تصویر ماژول عنوان شغلی

۹ استخراج حقوق مورد انتظار

در این ماژول ما با پیش پردازش مناسب، به دنبال کلماتی مانند «حقوق مورد نظر»، «حقوق مد نظر» و موارد از این دست می‌گردیم و در فاصله مناسب از این کلمات، عدد نوشته شده را به کاربر برمی‌گردانیم.

```python
import re


class ExpectedSalaryExtractor:
    def __init__(self) -> None:
        keys = [
            "حقوق مورد انتظار",
            "حقوق مورد انتظار",
            "حقوق مدنظر",
            "حقوق دلخواه",
        ]
        self.pattern_keys = r"({})\s*:*".format("|".join(keys))

    def find(self, text1):
        text = text1
        for match_key in re.finditer(self.pattern_keys, text):
            text2 = text[match_key.end():]
            if text2.find('\n') != -1:
                return text2[:text2.find('\n')]
            return text2
        return "None"
```

شکل ۸: تصویر ماژول حقوق مورد نظر

۱۰ استخراج نوع شغل مورد نظر

در این ماژول ابتدا پیش پردازش مناسب بر روی داده انجام می‌دهیم. سپس با جستوجو کلمات کلیدی مانند «نوع شغل مورد نظر»، «نوع شغل دلخواه» به دنبال مکان‌های احتمالی که در اطراف آنها نوع شغل مورد نظر کاربر نوشته شده باشد می‌گردیم و پس از آن در اطراف هر یک از آنها با جستوجو کلمات کلیدی مانند «پاره وقت»، «تمام وقت» و مواردی از این دست شغل مورد نظر کاربر را پیدا می‌کنیم.

```python
import re


class JobExpectedExtractor:
    def __init__(self) -> None:
        keys = [
            "نوع شغل موردنظر",
            "شغل موردنظر",
            "شغل دلخواه",
            "نوع شغل مورد نظر",
        ]
        values = [
            "تمام وقت",
            "تمام وقت",
            "پاره وقت",
            "کارآموزی",
        ]
        self.pattern_keys = r"({})\s*:*".format("|".join(keys))
        self.pattern_values = r"({})".format("|".join(values))

    def find(self, text1):
        text = text1
        text = re.sub(r"\n+", r" ", text)
        text = re.sub(r"\s+", r" ", text)
        for match_key in re.finditer(self.pattern_keys, text):
            for match_value in re.finditer(
                self.pattern_values, text[match_key.end() : match_key.end() + 40]
            ):
                return match_value.group(1)
        return "None"
```

شکل ۹: تصویر ماژول نوع شغل

۱۱ استخراج بخش‌های اضافه

برای بخش‌های اضافی، ابتدا لیستی از عناوین این بخش‌ها (شامل سوابق تحصیلی، سوابق شغلی، دستاوردها، پروژه‌ها و غیره) را در فایل keywords.txt قرار می‌دهیم. سپس، با خواندن آن‌ها شروع به بخش کردن رزومه به هر یک از آن‌ها می‌کنیم. در نهایت این بخش‌ها را تشخیص داده و خروجی می‌دهیم.

```python
class SectionExtractor:
    def __init__(self):
        with open('cv_info_extractor/resources/keywords.txt', 'r', encoding="utf-8") as file:
            self.keywords = [l.strip().split(',') for l in file.readlines()]

    def first_index(self, inp, keywords):
        for k in keywords:
            if inp.find(k) != -1:
                return inp.find(k)
        return -1

    def match_sections(self, inp):
        idxs = [{k[0]: self.first_index(inp, k)} for k in self.keywords if self.first_index(inp, k) != -1]
        idxs.append({'END': len(inp)})
        idxs.sort(key=lambda x: list(x.values())[0])
        idx_vals = [list(i.values())[0] for i in idxs]
        matches = [{list(idxs[i].keys())[0]: inp[idx_vals[i] + len(list(idxs[i].keys())[0]) + 1:idx_vals[i + 1]]} for i
                   in range(len(idx_vals) - 1)]
        return matches

    def find_sections(self, text):
        matched_sections = self.match_sections(text)
        if not matched_sections:
            return 'Not Found'
        return matched_sections
```

شکل ۱۰: تصویر ماژول استخراج بخش‌های اضافه

۱۲ نتایج

برای خروجی، کافی است که لیستی از رزومه‌هایی که می‌خواهیم را در فایل cv_extractor.py قرار دهیم. سپس با اجرای این فایل، خروجی به صورت CSV در فایل output.csv قرار می‌گیرد. در زیر نمونه‌ای از خروجی را که نتیجه حاصل از اجرای استخراج کننده بر روی دو رزومه است را مشاهده می‌کنید.

مهارت های نرم	وضعیت اشتغال	مهارتهای فنی	پروژه‌ها	نمرات مهم	سوابق شغلی
	تا کنون شاغل	JavaScript C SQL HTML / Assembly Ra PyTorch Red	آذر ۱۳۹۹ تاکنون ترمیکس پلتفرمی برای انتخاب واحد دانشجوی ان، دارای نسخه وب و اندروید. در دست راه انداز ی JavaScript, HTML, CSS, React. JS, Node. JS, Java, MySQL, MongoDB, Redis, Web Crawling , Android, SQLite, Gradle, Maven, GitHub فروردین ۱۳۹۹ تا مرداد ۱۳۹۹ پروژه درس برنامه‌سازی پیشرفته فروشگاه اینترنتی، متن باز، رتبه دوم پروژه مرتر نیمسال Java, JavaFX, SQLite, Socket Programming, Maven, GitHub CSV‌و‌Email مرداد ۱۳۹۹ تا شهریور ۱۳۹۹ برنامه‌های برای ارسال ایمیل‌های گروهی با پارامترهای مجزا برای هر فرد Java, JavaFX, HTML, SMTP اسفند ۱۳۹۹ تا فروردین ۱۴۰۰ پروژه درس آمار و احتمال مهندسی تشخیص هرزنامه در پست الکترونیک Python, NumPy, Pandas خرداد ۱۴۰۰ تاتیر ۱۴۰۰ پروژه درس طراحی زبانهای برنامه سازی Statement‌ها پیاده‌سازی مفسر برای یک زبان بر پایه Racket ۳	آمار و احتمال مهندسی ۲۰۰ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران جبر خطی ۱۹.۷ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران ساختمان داده‌ها و الگوریتم‌ها ۱۹.۸ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران طراحی زبانهای برنامه سازی ۱۹.۵ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران برنامه‌سازی وب ۲۰۰ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران ۳ برنامه‌سازی موبایل ۲۰۰ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران برنامه‌سازی پیشرفته ۱۹.۸ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران	شهریور ۱۴۰۰ تاکنون دستیار پژوهشی دانشگاه صنعتی شریف، تهران تحت نظارت جناب آقای دکتر رهیمی بهمن ۱۴۰۰ تاکنون سرپرست گروه دستیاران آموزشی دانشگاه صنعتی شریف، تهران درس جبر خطی: ارائه دهنده: جناب آقای دکتر رهیمی ۲ بهمن ۱۴۰۰ تاکنون دستیار آموزشی دانشگاه صنعتی شریف، تهران درس جبر خطی: ارائه دهنده: جناب آقای دکتر رهیمی شهریور ۱۴۰۰ بهمن ۱۴۰۰ دستیار آموزشی دانشگاه صنعتی شریف، تهران درس آمار و احتمال مهندسی، ارائه دهنده: جناب آقای دکتر مطهری شهریور ۱۴۰۰ تا بهمن ۱۴۰۰ دستیار آموزشی دانشگاه صنعتی شریف، تهران درس جبر خطی: ارائه دهنده: جناب آقای دکتر رهیمی، سرکار خانم رمضانی مهر ۱۳۹۹ تا اسفند ۱۳۹۹ سرپرست گروه دستیاران آموزشی دانشگاه صنعتی شریف، تهران درس مبانی برنامه‌سازی، زبان پایتون: ارائه دهنده: مهندس کدیور و مقدمات تحلیل داده NumPy و Pandas آموزش زبان پایتون به همراه کتابخانه‌های
کار تیمی انتقادپذیری قدرت تصمیم گیری مسئولیت پذیری مدیریت تیم رقابت جویی برنامه ریزی استراتژیک نوآوریزیان فارسی انگلیسی	علاقه مند		متن باز تا شهریور ۹۹۲۱مرداد یادنگار ۹۹۲۱ تا مرداد ۸۹۲۱فروردین فروشگاه انلاین ۹۹۲۱ تا بهمن ۸۹۲۱دی بهام رسان ۸۹۲۱ تا اردیبهشت ۸۹۲۱اردیبهشت بازی شطرنج‌باز پروژه برای اندروید توشته شده و اپلیکیشنی انلاین به عنوان یادنگار است که شبکه ی ۸۹۲۱ .اجتماعی کوچکی است که میتوانید لحظات خود را آن جا ثبت کنید این پروژه مربوط به درس برنامه نویسی پیشرفته است که فروشگاه انلاین با زبان java .است .که این پروژه حاصل کار تیمی بوده است و با معماری کلاینت سرور طراحی شده است پیام رسان با کمک زبان تکو با معماری سرور کلاینت بازی شطرنج گرافیکی با زبان جاوا	مبانی برنامه نویسی۲۰ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف برنامه نویسی پیشرفته۲۰ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف ساختمان گسسته۹۱.۷ کارشناسی مهندسی کامپیوتر، دانشگاه صنعتی شریف	

شکل ۱۱: تصویری از خروجی دو رزومه