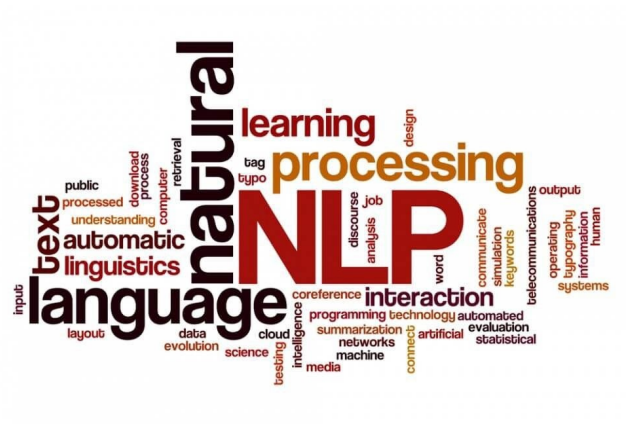
بسم الله الرحمن الرحیم



دانشکده مهندسی برق و کامپیوتر

پردازش زبان‌های طبیعی - تمرین اول

سید مهدی رضوی

استاد : آقای دکتر فیلی

اسفند ماه ۱۴۰۲

فهرست مطالب

فهرست تصاویر

۱ تمرین اول

۱. این زبان منظم متن را بر اساس **کلمه** توکن توکن خواهد کرد.

در زبان منظم b به عنوان مرز بین دو کاراکتر (whitespace) w تعریف می‌شود.

از این بابت مطمئن خواهیم بود که این جداکننده ، کلمات را بر اساس **کلمه** جدا خواهد کرد ، نه بر اساس زیرکلمه و یا کاراکتر.

همچنین با اجرای کد در بخش دوم از این بابت مطمئن خواهیم شد.

۲. طبیعتا از اصلی‌ترین ایراداتی که در tokenizer بالا مشاهده می‌شود ، در درجه نخست ناتوانی در نمایش کامل **تاریخ** و همچنین عدم نمایش کامل کلمه **M.Sc.** می‌باشد.

یعنی tokenizer متوجه نشده‌است که این دو کلمه یک کلمه مستقل هستند و نباید parse شوند.

این‌گونه می‌توان استدلال کرد که باید کاراکترهای whitespace را محدودتر نمود تا این زبان منظم هر کاراکتری را به عنوان whitespace نپذیرد.

۳. در زبان منظم اصلاح‌شده بعضی کاراکترهای معروف مثل **آپاستروف** ، **dash** ، **نقطه** و **backslash** را به عنوان whitespace در نظرنگرفته‌ایم.

در شکل ۲ می‌توانید کد آن را مشاهده بفرمایید.

```
sample = '- Just received my M.Sc. diploma today, on 2024/02/10! Exicted to embark on this new journey of knowledge and discovery.#MScGraduate #EducationMatters.'

custom_tokenizer(sample)

# بر حسب کلمه توکن ساخته است.
```

```
['Just',
 'received',
 'my',
 'M',
 'Sc',
 'diploma',
 'today',
 'on',
 '2024',
 '02',
 '10',
 'Exicted',
 'to',
 'embark',
 'on',
 'this',
 'new',
 'journey',
 'of',
 'knowledge',
 'and',
 'discovery',
 'MScGraduate',
 'EducationMatters']
```

شکل ۱: First Version of tokenizer and tokens

```
[2] import re
    def custom_tokenizer_modified(text):
        pattern = r"\b[\w'-\/.]+\b"  # Updated pattern to include backslashes and dashes apostrophes and hyphens
        tokens = re.findall(pattern, text, re.IGNORECASE)  # Added re.IGNORECASE flag for case insensitivity
        return tokens
```

```
sample = '- Just received my M.Sc. diploma today, on 2024/02/10! \
Exicted to embark on this new journey of knowledge and discovery.#MScGraduate #EducationMatters.\
 don\'t cat\'t Ahmad-Norrollahi'
custom_tokenizer_modified(sample)
```

```
['Just',
 'received',
 'my',
 'M.Sc',
 'diploma',
 'today',
 'on',
 '2024/02/10',
 'Exicted',
 'to',
 'embark',
 'on',
 'this',
 'new',
 'journey',
 'of',
 'knowledge',
 'and',
 'discovery',
 'MScGraduate',
 'EducationMatters',
 "don't",
 "cat't",
 'Ahmad-Norrollahi']
```

شکل ۲: Modified Version of tokenizer and tokens

۲ تمرین دوم

الگوریتم‌های WordPiece(BERT) و BPE(GPT) از معروف‌ترین الگوریتم‌ها برای نشانه‌گذاری کلمات هستند. این الگوریتم‌ها متن را به **زیرکلمه‌ها** نشانه‌گذاری می‌کنند.
در الگوریتم WordPiece با رویکرد ادغام حریصانه ، تلاش می‌کند تا **دنباله‌ای از کاراکترها** با بیشترین تکرار برای تشکیل واحدهای زیرکلمه ساخته‌شوند.
رویکرد این الگوریتم بر آن است که با توجه به واژگان احتمال داده‌های آموزشی به حداکثر خود برسد.
هنگام توکن کردن یک کلمه، الگوریتم ابتدا سعی می کند آن را به زیرکلمه های کامل از واژگان تبدیل کند. اگر کلمه ای را نتوان به طور کامل نشانه گذاری کرد، آن را به واحدهای فرعی کوچکتر تقسیم می کند. این به الگوریتم اجازه می دهد تا کلمات خارج از واژگان را با تجزیه آنها به واحدهای فرعی شناخته شده مدیریت کند.
یکی از مزیت‌های الگوریتم این است که می‌تواند کلمات نادر یا دیده نشده را با تجزیه آنها به واحدهای زیرکلمه که بخشی از واژگان هستند، مدیریت کند. این به بهبود تعمیم مدل های آموزش داده شده بر روی داده های متنی کمک می کند.

الگوریتم Byte Pair Encoding(BPE) : با **کاراکترهای منفرد** به عنوان واژگان اولیه شروع می شود و به طور مکرر متداول ترین جفت نمادها را برای ایجاد واحدهای بزرگ‌تر جدید مانند زیرکلمه ادغام می کند.
به همین علت این الگوریتم انعطاف‌پذیری بیشتری در ساختن کلمات و همچنین تنوع بیشتری در ساختن کلمات جدید دارد.
همچنین فرآیند ادغام کاراکترها تا رسیدن به یک معیارهمگرایی ادامه خواهد داشت . یعنی یک معیارهمگرایی ازپیش تعریف‌شده برای الگوریتم وجود دارد.

- رویکرد WordPiece از کاراکتر تنها شروع به ساختن زیرکلمه می‌کند اما رویکرد BPE از مجموعه کاراکتر متوالی برای شروع ساختن زیرکلمه استفاده می‌کند.
- استراتژی رویکرد WordPiece بر مبنای LikeLihood-based merging می‌باشد. اما استراتژی رویکرد BPE بر مبنای میزان تکرار و وقوع کاراکترها می‌باشد.
- در رویکرد BPE ما میزانی برای میزان همگرایی داریم که ازپیش تعریف شده است بر اساس تعداد کلمات vocabulary اما در رویکرد WordPiece بر اساس likelihood به ساختن زیرکلمات ادامه خواهیم‌داد.

در صورت استفاده از پکیج‌های پیش‌ساخته پایتون تعداد توکن‌های هر دوی این رویکردهای نشانه‌گذاری ۳۰۰۰۰ خواهد بود.

دلایل استفاده از این دو
برای مدل‌های زبانی بزرگ به صورت زیر خواهند بود :

- سازگاری با مدل‌های از پیش آموزش دیده مانند BERT , GPT
- مدیریت کارآمد توکن‌ها
- توکن‌کردن زیرکلمات
- مدیریت کلمات خارج از مجموعه لغات

۳ تمرین سوم

برای ساختن مدل N-gram و همچنین محاسبه احتمالات پیش‌آمد یک کلمه خاص بعد از یک مجموعه کلمه می‌توان از ماتریس دوبعدی استفاده کرد.
در این صورت با صفرهای زیادی روبرو خواهیم شد که راهکار مرسوم برای مقابله با آن استفاده از هموارساز add-k smoothing می‌باشد.
اما رویکردی که در این تمرین پیش‌گرفته‌ایم ، استفاده از ساختمان‌داده دیکشنری برای ذخیره تعداد ظاهر شدن یک کلمه بعد از یک مجموعه توالی خاص است .
در این صورت ما مشکل data sparcity نخواهیم داشت. همچنین رویکردی که در این تمرین پیاده‌سازی کرده‌ام با یک iteration تمام شمارش‌ها و محاسبات صورت می‌پذیرد. (پیچیدگی زمانی الگوریتم O(n) خواهد بود.) مجموعه رشته‌هایی که بعد از رشته معلوم توسط مدل تولید خواهدشد عبارت‌است از :
: bigram

Knowing well the windings of the trail he

come assembl compani fag princess brynilda wife affianc malud assert

: 3-gram

Knowing well the windings of the trail he

halfwit scare countri locat unit state world cost almost restrict

5-gram

Knowing well the windings of the trail he

halfwit scare countri locat unit state part world cost almost

در صورت استفاده از مدل n-gram برای n های خیلی بزرگ ، در صورت استفاده از ماتریس با مشکل حافظه روبرو خواهیم‌بود .
همچنین پیچیدگی زمانی الگوریتم نیز افزایش خواهد یافت.
در صورت اسفاده از رویکردی که ما در این تمرین ارائه کردیم نیز همانطورکه در نتایج از ۳ به ۵ مشاهده‌می‌شود ، میزان بهبود به صورت مشهود نبوده است و میزان رخدادها به مراتب کمتر خواهدشد. (میزان بهبود در دقت عملکرد مدل چشمگیر نخواهد بود.)

۴ تمرین چهارم

n : 2
Precision: 1.0
Recall: 0.20754716981132076
F1 Score: 0.34375000000000006

n : 3
Precision: 0.5925925925925926
Recall: 0.3018867924528302
F1 Score: 0.4

n : 4
Precision: 0.5714285714285714
Recall: 0.3018867924528302
F1 Score: 0.3950617283950617

n : 5
Precision: 0.5714285714285714
Recall: 0.07547169811320754
F1 Score: 0.13333333333333333

n : 6
Precision: 0.5
Recall: 0.03773584905660377
F1 Score: 0.07017543859649122

همانطور که از نتایج مشخص است ، با افزایش میزان n به طور محسوسی میزان امتیاز F1 در حال کاهش است . یعنی مدل در حال ضعیف‌تر شدن است.
تابع تست نیز بر اساس برچسب‌های خود داده و بر اساس پرکردن عناصر ماتریس آشفتگی خواهدبود. (True Positive , True Negative , False Positive , False Negative)