بسم الله الرحمن الرحیم

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پردازش زبان طبیعی

نیم‌سال دوم ۰۲-۰۱

مدرس: احسان‌الدین عسگری

تمرین دوم — **استخراج اطلاعات قاعده‌محور با عبارات منظم** — مهلت ارسال: ۱۶ فروردین

- مهلت ارسال پاسخ تا ساعت ۲۳:۵۹ روز مشخص شده است.
- در تمرین‌هایی که چند چالش دارند، فقط یک نفر از هر گروه در سامانه CW باید چالش مورد نظر گروه را انتخاب کند. امکان تغییر چالش تا قبل از زمان ددلاین انتخاب چالش وجود دارد. البته ذکر این نکته ضروری است که هر چالش محدودیتی برای تعداد افرادی که آن را انتخاب می‌کنند، دارد. بنابرین در اسرع وقت برای انتخاب چالش اقدام کنید.
- در طول ترم امکان ارسال با تاخیر تمرین‌ها بدون کسر نمره تا سقف ۱۲ روز وجود دارد. محل بارگزاری جواب تمرین‌ها مطابق زمان مشخص شده در تقویم، بسته خواهد شد و پس از گذشت این مدت، پاسخ‌های ارسال‌شده پذیرفته نخواهند شد. همچنین، به ازای هر روز تأخیر غیر مجاز ۱۵ درصد از نمره تمرین کسر خواهد شد.
- توجه داشته‌باشید که نوت‌بوکهای شما باید قابلیت بازاجرای ۱۰۰ درصد داشته باشند و در صورت نیاز به نصب یک کتابخانه یا دسترسی به یک فایل، مراحل نصب و دانلود (از یک محل عمومی) در نوت‌بوک وجود داشته باشد.
- تمامی فایل‌های مرتبط به پروژه که حجم کمی دارند باید به شکل فایل زیپ در سامانه CW اپلود شوند. اگر حجم یک فایل زیاد بود (مانند فایل ذخیره شده یک مدل در صورتیکه بیش از ۲۰۰ مگابایت باشد)، تنها همان فایل را در یک محل عمومی، مثل گوگل درایو آپلود بفرمایید و لینک دانلود را در نوت‌بوک و مستندات قرار دهید.
- در پروژه‌های گروهی کافی است که فقط یکی از اعضای گروه پروژه را آپلود کند. اما حتما در گزارش کار نام همه اعضای گروه همراه با شماره دانشجویی آن‌ها آورده شود.
- بخشی از نمره شما به گزارش کار شما اختصاص دارد. در گزارش کار لازم نیست خط به خط کاری را که کرده‌اید توضیح دهید. بلکه باید به شکل کلی ایده‌تان برای حل مساله را شرح دهید. لازم است چند نمونه از خروجی‌های مساله را در گزارش بیاورید و براساس آن رفتار برنامه‌تان را تحلیل کنید. همچنین اگر پارامتری در صورت مساله خواسته شده (مانند دقت، صحت و مواردی از این دست) که در گزارش آورده شود شما باید آن را حساب کنید و در گزارش خود بیاورید.
- کد نهایی شما باید یک کلاس شامل تابع run داشته باشد که برای هر چالش طبق ورودی‌های مشخص شده، خروجی موردنظر را بدهد. شما می‌توانید به کمک روند معرفی شده در فایل CONTRIBUTION کدهای خود را به کتابخانه‌ی parsi.io اضافه کنید. در صورتی که pull-request شما پذیرفته شود نمره‌ی امتیازی به شما تعلق خواهد گرفت.
- دقت داشته باشید، موارد امتیازی که در این تمرین آمده است، صرفا بر روی امتیاز همین تمرین اثر دارد و بر روی نمرات تمارین و یا بخش‌های دیگر درس، تاثیر ندارد.
- در صورت وجود هرگونه ابهام یا مشکل، در کوئرای درس آن مشکل را بیان کنید و از پیغام دادن مستقیم به تیم تدریس خودداری کنید.

درست کردن هۀ کسره

هدف از این بخش، اصلاح اشتباه‌های رایج در کاربرد هۀ کسره و نقش نمای اضافه است. زبان فارسی تعداد زیادی تکواژ با صدای e دارد. همین موضوع باعث می‌شود تا در استفاده از «ـِ» و «ه» دچار اشتباه شویم. در نگارش فارسی، کسره «ـِ» و حرف «ه» به یک صورت تلفظ می‌شوند و به آن هۀ کسره می‌گویند. رعایت نکردن هۀ کسره، مشکل رایجی است که در متون پیام‌ها و نظرات، بسیار رخ می‌دهد.

در این بخش از شما انتظار می‌رود که سامانه‌ای توسعه دهید که به عنوان ورودی، یک متن فارسی دریافت کند و متنی فارسی‌ای را که ایرادات هۀ کسره‌اش تصحیح شده‌است، به همراه بازه‌ای که ایرادات در آن‌ها قرار دارند، به عنوان خروجی تولید کند. در جدول زیرچند مثال برای درک بهتر آورده شده است.

ورودی	خروجی
حاله من خیلی خوب.	`{ "correct": ".حال من خیلی خوبه", "حاله": [0, 4], "خوب": [13, 16] }`
می‌خواهم پرنده از شاخه، روی بام خانه بپره. بپر.	`{ "correct": ".می‌خواهم پرنده از شاخه، روی بام خانه بپره", "بامه": [27, 31], "بپر": [37, 40] }`
دیروز آن مرده قوی مرد.	`{ "correct": ".دیروز آن مرد قوی مرد", "مرده": [9, 13] }`

ایجاد جمله‌جمله‌کننده و کلمه‌کلمه‌کننده‌ی متون[1]

یکی از ملزومات پردازش زبان طبیعی، پیش‌پردازش‌های ابتدایی متن است. از این پیش‌پردازش‌ها می‌توان به جمله‌جمله‌کردن و کلمه‌کلمه‌کردن‌متون اشاره کرد. یعنی محدوده‌ی هر کلمه و هر جمله به درستی تشخیص داده شود.

ابزارهای متنوعی برای پیش‌پردازش زبان فارسی وجود دارد. کتابخانه‌ی هضم از نمونه‌ی ابزارهای متن‌باز است. در این تمرین شما لازم است در این تمرین، انتظار می‌رود قسمت‌های SentenceTokenizer و یا WordTokenizer از کتابخانه‌ی هضم یا جلمه‌جلمه کنندگان و کلمه‌کلمه کنندگان موجود در سایر کتابخانه‌ها را بررسی کنید، نقاط ضعف و قوت آن‌ها را به صورت test-case مشخص کنید. در قدم بعد سعی کنید تعدادی از نقاط ضعفی که مشخص کردید را بهبود ببخشید. توجه کنید که مستقل از اینکه کتابخانه‌ی اولیه‌ای که بر روی آن کار می‌کنید نسخه‌ی نهایی کد شما باید بر اساس ساختار کتابخانه‌ی parsi.io باشد.

در صورتی که توسعه‌ی شما باعث تغییر در نسخه‌های آتی این کتابخانه شود (یعنی pull-request تان توسط نگهدارنده‌های parsi.io قبول شود)، نمره‌ی امتیازی برای این تمرین به تیمتان تعلق می‌گیرد.

برای انجام این تمرین، فازهای پیشنهادی به صورت زیر است:

۱. مثال‌هایی پیدا کنید که نسخه‌ی فعلی هضم روی آن‌ها به درستی جمله‌جمله کردن و یا کلمه‌کلمه کردن متن را انجام نمی‌دهد. هر چه دامنه‌ی این مثال‌ها در متون رایج فارسی، گسترده‌تر باشد، بهتر است.

۲. کد هضم برای قسمت‌های WordTokenizer و یا SentenceTokenizer را بخوانید و با مثال‌هایی که می‌زنید، جاهایی در کد که امکان بهبود دارد را پیدا کنید.

۳. دقت کد خود را بر روی مثال‌هایی که پیدا کردید بررسی کنید. هر چه بهبودهایی که می‌دهید بیشتر باشند و حالت‌های کلی‌تری را شامل شود، بهتر است. با توجه به این که تمرکز این تمرین روی عبارات منظم هم هست، سعی کنید در تغییراتتان از re استفاده کنید.

۴. روی یک متن دلخواه، عملکرد جمل‌جلمه کردن و کلمه‌کلمه کردن متن را برای قبل و بعد از تغییرات، مقایسه کنید. در کل انتظار داریم عملکرد، بهتر شده باشد. این مقایسه را برای حالت کلی زبان فارسی هم انجام دهید (نه فقط مثال‌هایی که در فاز ۱ پیدا کردید).

---

[1] sentenceTokenizer or sentencizer and tokenizer

ایجاد نرمال‌ساز[۲]

یکی از ملزومات پردازش زبان طبیعی، پیش‌پردازش‌های ابتدایی متن است. از این پیش‌پردازش‌ها می‌توان به نرمال‌سازی متون اشاره کرد. یعنی یک شیوه‌ی استانداردی برای نوشتن باشد و ما متون را تا جای ممکن به آن شبیه کنیم. برای مثال می‌توان به تبدیل کاراکتر «ي» عربی به «ی» فارسی یا گذاشتن نیم‌فاصله برای افعالی که «می» دارند، اشاره کرد.

ابزارهای متنوعی برای پیش‌پردازش زبان فارسی وجود دارد. کتابخانه‌ی هضم از نمونه‌ی ابزارهای متن‌باز است. در این تمرین انتظار می‌رود، قسمت‌های Normalizer و یا InformalNormalizer از کتابخانه‌ی هضم یا نرمال‌ساز موجود در سایر کتابخانه‌ها را بررسی، نقاط ضعف و قوت آن‌ها را به صورت test-case مشخص کنید. در قدم بعد سعی کنید تعدادی از نقاط ضعفی که مشخص کردید را بهبود ببخشید. توجه کنید که مستقل از اینکه کتابخانه‌ی اولیه‌ای که بر روی آن کار می‌کنید نسخه‌ی نهایی کد شما باید بر اساس ساختار کتابخانه‌ی parsi.io باشد.

در صورتی که توسعه‌ی شما باعث تغییر در نسخه‌های آتی این کتابخانه شود (یعنی pull-request تان توسط نگهدارنده‌های parsi.io قبول شود)، نمره‌ی امتیازی برای این تمرین به تیمتان تعلق می‌گیرد.

برای انجام این تمرین، فازهای پیشنهادی به صورت زیر است:

۱. مثال‌هایی پیدا کنید که نسخه‌ی فعلی هضم روی آن‌ها به درستی نرمال‌سازی کملات یا جملات را انجام نمی‌دهد. هر چه دامنه‌ی این مثال‌ها در متون رایج فارسی، گسترده‌تر باشد بهتر است.

۲. کد هضم برای قسمت‌های Normalizer و یا InformalNormalizer را بخوانید و با مثال‌هایی که می‌زنید، جاهایی در کد که امکان بهبود دارد را پیدا کنید.

۳. دقت کد خود را بر روی مثال‌هایی که پیدا کردید بررسی کنید. هر چه بهبودهایی که می‌دهید بیشتر باشند و حالت‌های کلی‌تری را شامل شود، بهتر است. با توجه به این که تمرکز این تمرین روی عبارات منظم هم هست، سعی کنید در تغییراتتان از re استفاده کنید.

۴. روی یک متن دلخواه، عملکرد نرمال‌سازی را برای قبل و بعد از تغییرات، مقایسه کنید. در کل انتظار داریم عملکرد، بهتر شده باشد. این مقایسه را برای حالت کلی زبان فارسی هم انجام دهید (نه فقط مثال‌هایی که در فاز ۱ پیدا کردید)

---

[۲] Normalizer

تشخیص کلمات غیرقانونی

هدف از این بخش، تشخیص کلمات غیرقانونی فارسی است که ممکن است با طرق خاصی تغییر کرده باشند. بات‌هایی[۳] وجود دارند که وظیفه‌شان تشخیص کلمات غیر قانونی است، ولی بعضی از این بات‌ها نمی‌توانند این نوع کلمات را زمانی که بین حروفشان کاراکترهای غیرمرتبط بیاید، تشخیص دهند.

در این تمرین، وظیفه‌ی شما این است که برای این بات‌ها سامانه‌ای توسعه دهید که بتوانند کلمات غیرقانونی‌ای را تشخیص دهد که ممکن است بین حروفشان، حروف غیر فارسی، از جمله حروف انگلیسی، اعداد و کاراکترهای خاص و ... آمده باشد.

در این سوال تابع run شما باید علاوه بر رشته‌ی ورودی، یک لیستی از کلمات غیرقانونی‌ای که باید در رشته‌ی ورودی، پیدا کند را دریافت کند.

```
run(input: str, illegal_words: list)
```

ورودی شما یک رشته متن فارسی، به همراه لیستی از کلمات غیرقانونی خواهد بود و شما باید در خروجی، این کلمات غیرقانونی را که با حروف اضافه در متن آمده‌اند را به همراه بازه‌ای که این کلمات در آن قرار دارد دارند را تولید کنید. در جدول زیر چند نمونه از ورودی و خروجی، به عنوان نمونه و برای درک بهتر آورده شده است.

ورودی	خروجی
Input = "این &تف...٪ن۸گ# را فروختم" Illegal_words = ["تفنگ"]	{ "تفنگ": [4, 14] }
input="با ق*bla شق و fچ^!نگ4ال غذا خوردم." illegal_words = ["قاشق", "چنگال"]	{ "قاشق": [3, 10], "چنگال": [14, 23] }

[۳] Bots

پیدا کردن زیرجملات و کلمات ربطی

یکی از موارد کاربردی در پردازش متون، پیدا کردن قسمت‌های مختلف جمله است. خیلی از جملاتی که در فارسی رایج نوشته می‌شوند به صورت مرکب هستند. یعنی تعدادی زیرجمله دارند که با کلمات ربط مثل «و»، «یا»، «اما» و ... یا بعضا بدون هیچ کلمه‌ی ربطی، از هم جدا شده‌اند.

در این قسمت، در ورودی یک متن داده می‌شود و در خروجی انتظار داریم زیرجمله‌ها و کلمات ربط بین‌شان از هم جدا شوند و انواع حروف ربط، مانند حروف ربط هم‌پایه‌ساز، وابسته‌ساز، و همچنین حروف ربط تضاد و مزدوج را تشخیص دهید. لطفا به این لینک برای اطلاعات بیشتر در مورد حروف ربط مراجعه کنید.

در کد نهایی باید یک تابع داشته باشید که در ورودی یک رشته (string) به عنوان متن بگیرد و در خروجی یک لیست از کلمات ربطی و زیرجمله‌ها و در کنار آن، تمام حروف ربط به کار برده شده را به همراه نوعشان و بازه‌ای که در جمله قرار دارند، به صورت یک دیکشنری پایتون تولید کنید.
درصورتی که چند بار از یک حرف ربط در جمله‌ی ورودی استفاده شده، با قرار دادن شماره در کنار آن‌ها تمایز ایجاد کنید، برای مثال آن حرف ربطی که زودتر آمده، شماره‌ی کمتری دارد.

تعدادی مثال در جدول زیر آورده شده‌است:

ورودی	خروجی
امروز در خبرها آمده است که هوای شهر نیمه‌ابری است.	`{ "splits": ["امروز در خبرها آمده است","که","هوای شهر نیمه ابری است"], "که": ["وابسته‌ساز", [24, 26]] }`
او درس خواند اما نمره‌اش خوب نشد ولی کلی یاد گرفت.	`{ "splits": ["او درس خواند","اما","نمره‌اش خوب نشد","ولی","کلی یاد گرفت"], "اما": ["هم‌پایه‌ساز (تضاد)", [13, 16]], "ولی": ["هم‌پایه‌ساز (تضاد)", [32, 35]] }`
رفت به دوستش گفت بیا با هم بازی کنیم.	`{ "splits": ["رفت","به دوستش گفت","بیا","با هم بازی کنیم"] }`
هم فال بود هم تماشا.	`{ "splits": ["تماشا", "هم", "فال بود", "هم"], "هم(1)": ["هم‌پایه‌ساز (مزدوج)", [0, 2]], "هم(2)": ["هم‌پایه‌ساز (مزدوج)", [11, 13]] }`

استخراج ویژگی‌های سفارش غذا از متن پیام

در این بخش، هدف استخراج ویژگی‌های سفارش غذا، از متن پیام است. یک سامانه هوشمند ثبت سفارش غذا را در نظر بگیرید که به کمک آشپزان آمده است و از سفارش متنی مشتریان، غذای مورد نظر، به همراه ویژگی‌هایش را استخراج می‌کند. ویژگی‌هایی باید استخراج شوند که مربوط به کل غذا یا بخشی از غذا باشد. قاعدتا هر چه بتوانید ویژگی‌های بیشتری استخراج کنید و تمام قواعد مرتبط با آن‌ها را شناسایی نمایید، بهتر است.

ورودی شما یک رشته متن سفارش غذا به زبان فارسی خواهد بود و شما باید در خروجی، خود غذا را مشخص کنید و ویژگی‌های مرتبط با بخش‌های مختلف غذا را به صورت یک دیکشنری زبان پایتون تولید کنید، به طوی که کلید دیکشنری، آن بخش مشخص غذا باشد و مقدار آن، لیستی از ویژگی‌هایی باشد که آن بخش از غذا دارد. در جدول زیرچند مثال برای درک بهتر آورده شده است.

ورودی	خروجی

بی زحمت یک سبزی پلو با ماهی میخوام که لطفا سبزی اش کم باشد و اگر آب دریا خوب بوده است، ماهی قزل‌آلا بگذار اگر سبزی تان خوب است، سبزی پاک‌شده بگذار.	<pre>{
	"food": "سبزی پلو با ماهی",
	"سبزی": ["کم", "پاک‌شده"],
	"ماهی": ["قزل آلا"]
	}</pre>

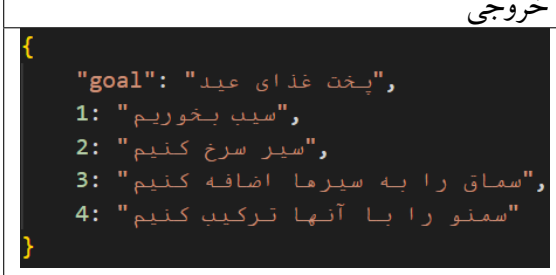
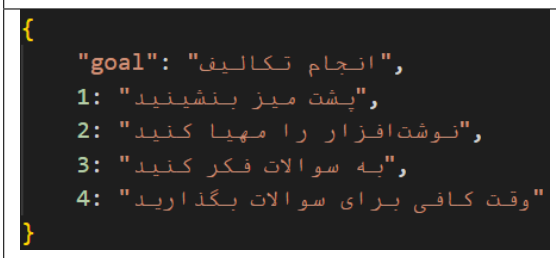
سلام، یک پیتزای قارچ و گوشت می‌خواستم و لطفا قارچ‌ها خوب شسته شده باشد و گوشت چرخ کرده و زیاد باشد.	<pre>{
	"food": "پیتزای قارچ و گوشت",
	"قارچ": ["خوب شسته شده"],
	"گوشت": ["چرخ کرده", "زیاد"]
	}</pre>

تشخیص توالی انجام یک کار

در بسیاری از مواقع، لازم است که توالی انجام کارها را از متون بدست آوریم، مانند اینکه از متن توضیحات یک سرآشپز در مورد طبخ یک غذا، دستور پخت آن غذا، توالی و ترتیب گام‌های مختلف را بدون هرگونه توضیحات اضافه‌ای، بدست آوریم.

در این تمرین از شما می‌خواهیم که یک سامانه‌ای را برای تشخیص ترتیب انجام کاری پیاده‌سازی کنید، به نحوی که آن کار به خصوص را تشخیص دهید و سپس توالی گام‌هایی که باید برای به انجام رساندن آن کار، برداشته شود را نیز پیدا کنید.

به عنوان ورودی، متنی به زبان فارسی به شما داده می‌شود و شما باید به عنوان خروجی یک دیکشنری زبان پایتون را که شامل کار هدف، و همچنین توالی انجام کارها به همراه شماره گذاری‌شان است، تولید کنید.
در جدول زیر چند نمونه برای درک بهتر آورده شده‌است.

ورودی	خروجی
برای پخت غذای عید، لازم است سیر سرخ کنیم ولی قبلش باید سیب بخوریم، در گام سوم باید سماق را به سیرها اضافه کنیم و بعد سمنو را با انها ترکیب می‌کنیم.	
جهت انجام تکالیف، ابتدا پشت میز بنشینید، سپس به سوالات فکر کنید اما پیش از آن، نوشت‌افزار را مهیا کنید تا بهتر تمرکز کنید در نهایت وقت کافی برای حل سوالات بگذارید.	

استخراج کلمات انگلیسی نوشته شده به فارسی

در این بخش، هدف استخراج کلمات انگلیسی نوشته شده به فارسی، در جمله‌ی ورودی است. در متون فارسی موجود در شبکه‌های اجتماعی، پست‌ها و نظرات، کلماتی وجود دارند که در واقع انگلیسی هستند ولی نوشتارشان به زبان فارسی است.

در گام پیش‌پردازش متون فارسی، تشخیص این گونه کلمات الزامی است، تا مطابق با مساله، اقدامات مناسبی مانند حذف کردن آن‌ها، صورت بگیرد.

در این تمرین از شما انتظار می‌رود که سامانه‌ای را توسعه دهید که ورودی‌اش یک رشته متن فارسی خواهد بود و به عنوان خروجی، کلماتی را که انگلیسی هستند ولی به فارسی نوشته شده‌اند را به همراه بازه‌ای که آن کلمات در رشته‌ی ورودی قرار دارند، تولید کند.
همچنین می‌توانید از این دیتاست که شامل لیستی از کلمات فارسی است یا از واژگان ویکی پدیای فارسی، استفاده نمایید.

در جدول زیر چند مثال برای درک بهتر خواسته‌ی این چالش، آورده شده است.

ورودی	خروجی

امروز یک کار خیلی هاردی داشتیم، ولی تو کانتریبیوشن خوبی داشتی، تنکس.	<pre>{
	"هارد": [18, 22],
	"کانتریبیوشن": [40,51],
	"تنکس": [64, 68]
	}</pre>

سیستم کامپیوترم خراب شده‌است.	<pre>{
	"سیستم": [0, 5],
	"کامپیوتر": [6,14]
	}</pre>

درخواست تعریف چالش جدید برای تمرین ۲

در صورتی که پیشنهاد جدیدی دارید و یا مایل هستید که روی بهبود یکی از عملکردهای فعلی parsi.io کار کنید، مشابه توضیحات و تعریف این چالش‌ها، پروپوزال یک صفحه‌ای بنویسید و در صورت تصویب روی آن کار کنید.
فایل پروپوزال گروه خودتان را، در محلی که برای تعریف پروپوزال برای تمرین ۲ در CW در نظر گرفته شده است، بارگزاری کنید.