به نام خدا

دانشکده علوم و فنون نوین

آشنایی با زبان‌شناسی رایانشی (۱۲۱-۰۵-۸۳)
نیم‌سال دوم ۱۴۰۱-۱۴۰۰
تمرین شماره ۴

تاریخ تحویل: ۱۴۰۱/۰۳/۳۰

---

۱. (۱۰٪) [**پژوهش: تجزیه نحوی وابستگی**] همانطور که می‌دانید یکی از انواع روش‌های تجزیه نحوی، تجزیه وابستگی (Dependency Parsing) است. در این تمرین مروری بر کاربردها و روش‌های این نوع تجزیه ارائه دهید.

---

۲. (۱۵٪) [**برچسب‌زنی اجزای کلام با HMM- دستی**] مدل مخفی مارکوف زیر را در نظر بگیرید.

	فعل	اسم	صفت	ضمیر
احتمال اولیه	۰.۲	۰.۵	۰.۱	۰.۲
فعل	۰.۰	۰.۶	۰.۲	۰.۲
اسم	۰.۴	۰.۱	۰.۴	۰.۱
صفت	۰.۳	۰.۲	۰.۲	۰.۳
ضمیر	۰.۱	۰.۶	۰.۲	۰.۱

	فعل	اسم	صفت	ضمیر
آن	۰.۰	۰.۰	۰.۰۱	۰.۱
مرد	۰.۰۲	۰.۱۵	۰.۱	۰.۰
کرد	۰.۱	۰.۰۵	۰.۰۱	۰.۰
خوشحالی	۰.۰۳	۰.۰	۰.۰۲۵	۰.۰

الف) ساختار مدل مخفی مارکوف فوق را رسم کنید و پارامترهای مدل را تعیین کنید.

ب) با محاسبه دستی و استفاده از مدل HMM فوق، برچسب‌های اجزای کلام مناسب را برای دنباله کلمات «**آن مرد کرد خوشحالی کرد**» را با الگوریتم ویتربی محاسبه کنید. توجه کنید که کلیه محاسبات را ارائه کنید.

h.veisi@ut.ac.ir

۳. (۶۰٪) **[پیاده‌سازی: برچسب‌زنی اجزای کلام با HMM]** هدف این تمرین پیاده‌سازی برچسب‌زن اجزای کلام برای زبان فارسی با استفاده از روش HMM و الگوریتم ویتربی است. در این تمرین از داده‌های ارائه شده به همراه تمرین استفاده کنید. داده‌های داده شده شامل یک فایل آموزش با اسم POST-Persian-Corpus-Train.txt (حدود ۳۰۰۰ جمله) و یک فایل آزمون با اسم POST-Persian-Corpus-Test.txt (حدود ۵۵۰ جمله) است که تمام کلمات آن برچسب‌گذاری دستی (با تعداد ۲۱ برچسب) شده‌اند. برای ساخت مدل HMM از داده فایل آموزش و برای ارزیابی کار خود و تست کردن کیفیت سیستم ساخته شده، از داده فایل آزمون استفاده شود. قبل از نوشتن توابع مرتبط با این تمرین، ابتدا دو فایل داده شده (آموزش و آزمون) را با هم ترکیب کرده و از روی آن لیست کلمات یکتا را استخراج کنید و از آن به عنوان واژگان استفاده کنید.

برای این تمرین در فاز آموزش سه تابع زیر را بنویسید:

- CalculateInitProb: این تابع پیکره آموزش و لیست برچسب‌ها را می‌گیرد و احتمال اولیه برچسب‌ها را محاسبه کرده و در یک فایل به نام ProbsPi.txt ذخیره می‌کند.
- CalculateTransProb: این تابع پیکره آموزش و لیست برچسب‌ها را می‌گیرد و احتمال انتقال برچسب‌ها را محاسبه کرده و در یک فایل به نام ProbsA.txt ذخیره می‌کند.
- CalculateEmisProb: این تابع پیکره آموزش، لیست برچسب‌ها و لیست همه کلمات واژگان را می‌گیرد و احتمال هر کلمه به شرط هر برچسب را محاسبه کرده و در یک فایل به نام ProbsB.txt ذخیره می‌کند.

برای فاز آزمون، یک کپی از فایل آزمون درست کنید (با اسم POST-Persian-Corpus-Test-Sent.txt) که در آن برچسب‌ها حذف شده باشد و از آن به عنوان داده تست استفاده کنید. در این مرحله دو تابع زیر را بنویسید:

- MyPOSTagger: این تابع احتمال‌های محاسبه شده در فاز آموزش، لیست برچسب‌ها و فایل

h.veisi@ut.ac.ir

به نام خدا

آزمون بدون برچسب (POST-Persian-Corpus-Test-Sent.txt) را گرفته و هر جمله را جدا کرده (پایان یافته با نقطه) و آن را برای تابع MyViterbi که کار آن برچسب زدن یک جمله با الگوریتم ویتربی است، می‌فرستد. خروجی تابع MyViterbi توسط تابع MyPOSTagger نگهداری شده و در پایان یک فایل متنی با اسم POST-Persian-Corpus-Test-MyOut.txt تولید می‌کند که جلو هر کلمه، برچسب پیشنهاد شده توسط سیستم شما نیز وجود دارد.

- CalculatePOSTAccuracy: این تابع دو فایل داده آزمون حاوی برچسب‌های درست (POST-Persian-Corpus-Test.txt) و حاوی برچسب‌های بدست آمده توسط سیستم شما (POST-Persian-Corpus-Test-MyOut.txt) را گرفته و با مقایسه برچسب‌های این دو فایل، درصد برچسب‌های درست تشخیص داده شده را به عنوان خروجی برگرداند.

---

۴. (۱۵٪) [**تجزیه نحوی با روش CKY**] با توجه به قوانین و واژگان داده شده در زیر، درخت(های) تجزیه جمله "**بچه شیر را خورد**" را بدست آورید.

$S \rightarrow NP \quad VP$	$VB \rightarrow$ خورد
$S \rightarrow NP \quad PA \quad VP$	$NN \rightarrow$ بچه
$VP \rightarrow VB \quad NP$	$NN \rightarrow$ شیر
$VP \rightarrow VB$	$AD \rightarrow$ شیر
$NP \rightarrow DT \quad NN$	$NN \rightarrow$ خورد
$NP \rightarrow NN$	$PA \rightarrow$ را
$NP \rightarrow AD \quad NN$	$AD \rightarrow$ بچه

---

h.veisi@ut.ac.ir

به نام خدا

دانشکده علوم و فنون نوین

آشنایی با زبان‌شناسی رایانشی (۱۲۱-۰۵-۸۳)

نیم‌سال دوم ۱۴۰۱-۱۴۰۰

تمرین شماره ۴

تاریخ تحویل: ۱۴۰۱/۰۳/۳۰

در این تمرین فرض شده است از تمرین‌های قبل، توابع زیر را اختیار دارید.

برخی از توابع تمرین‌های شماره ۱، ۲ و ۳
NoAllWords , NoAllUniqWords, NoEachUniqWord = **CountWordsFile**(TextFileName) تابعی که فایل متنی (TextFileName) را گرفته و تعداد کل کلمات (NoAllWords)، تعداد کل کلمات یکتا (NoAllUniqWords) و تکرار هر کدام از کلمات یکتا (NoEachUniqWord) را به عنوان خروجی برگرداند.
NoAllWordsPerFolder , NoAllUniqWordsPerFolder, NoEachUniqWordPerFolder, NoEachUniqWord = **CountWordsFolder**(Path) تابعی که مسیر (Path) حاوی یک یا چند پوشه را گرفته و تعداد کل کلمات هر پوشه (NoAllWordsPerFolder)، تعداد کل کلمات یکتا در هر پوشه (NoAllUniqWordsPerFolder)، تکرار هر کدام از کلمات یکتا در هر پوشه (NoEachUniqWordPerFolder) و تکرار هر کدام از کلمات یکتا در کل پوشه‌های موجود در مسیر داده شده (NoEachUniqWord) را به عنوان خروجی برگرداند.
OutFile = **CombineFiles** (Path) تابعی که مسیر (Path) را گرفته و تمام فایل‌های موجود در آن را با یکدیگر ادغام کند به گونه‌ای که هر سطر فقط حاوی مطالب یکی از فایل‌ها باشد. خروجی این تابع در متغیر OutFile قرار داده می‌شود.
NoChar = **CountChar** (InTextFileName, CharName) تابعی که فایل متنی (InTextFileName) و یک کاراکتر (CharName) را گرفته و تعداد کل آن کارکتر (NoChar) را در آن فایل به عنوان خروجی برگرداند.
OutTextFileName = **CorrectPuncs** (InTextFileName) تابعی که فایل متنی (InTextFileName) را گرفته و پس از اصلاح اشکال‌های علائم سجاوندی در متن، فایل خروجی (OutTextFileName) را به عنوان خروجی برگرداند.
OutTextFileName = **CorrectCoding** (InTextFileName, TableCodings) تابعی که فایل متنی (InTextFileName) و فایل حاوی اصلاحات (TableCodings) را گرفته و پس از اصلاح اشکال‌های کدگذاری در متن، فایل خروجی (OutTextFileName) را به عنوان خروجی برگرداند.
OutTextFileName = **CorrectVariations** (InTextFileName, TableVariations) تابعی که فایل متنی (InTextFileName) و فایل حاوی اصلاحات (TableVariations) را گرفته و پس از اصلاح اشکال‌های تنوع املایی در متن، فایل خروجی (OutTextFileName) را به عنوان خروجی برگرداند.
BiGramModel = **CalculateBiGram** (Corpus, Lexicon) تابعی که فایل متنی حاوی کل پیکره (Corpus) و فایل حاوی واژگان (Lexicon) را گرفته و احتمال یک‌تایی (Uni-gram) کلمات واژگان را محاسبه و در فایل خروجی (MonoGramModel) ذخیره می‌کند.
MonoGramModel = **CalculateMonoGram** (Corpus, Lexicon) تابعی که فایل متنی حاوی کل پیکره (Corpus) و فایل حاوی واژگان (Lexicon) را گرفته و احتمال دوتایی (Bi-gram) کلمات واژگان را محاسبه و در فایل خروجی (BiGramModel) ذخیره می‌کند.
Perplexity = **CalculatePerplexity** (TestFileName, NGramModel) تابعی که فایل متنی (TestFileName) حاوی جملات آزمون و فایل حاوی مدل زبانی (NGramModel) را گرفته و مقدار سرگشتگی را علاوه بر ذخیره در یک فایل، به عنوان یک عدد (Perplexity) برمی‌گرداند.

h.veisi@ut.ac.ir

به نام خدا

دانشکده علوم و فنون نوین

آشنایی با زبان‌شناسی رایانشی (۱۲۱-۰۵-۸۳)
نیم‌سال دوم ۱۴۰۱-۱۴۰۰
تمرین شماره ۴

تاریخ تحویل: ۱۴۰۱/۰۳/۳۰

بعد از این تمرین، تابع‌های زیر به مجموعه برنامه‌های شما افزوده می‌شود.

تابع تمرین شماره ۴
ProbsPi =**CalculateInitProb**(TrainSet, TagSet) این تابع داده آموزش و لیست برچسب‌ها را می‌گیرد و احتمال اولیه برچسب‌ها را در یک فایل به نام ProbsPi.txt ذخیره می‌کند.
ProbsA =**CalculateTransProb**(TrainSet, TagSet) این تابع داده آموزش و لیست برچسب‌ها را می‌گیرد و احتمال انتقال برچسب‌ها را در یک فایل به نام ProbsA.txt ذخیره می‌کند.
ProbsB =**CalculateEmisProb**(TrainSet, TagSet, Lexicon) این تابع پیکره آموزش، لیست برچسب‌ها و واژگان را می‌گیرد و احتمال هر کلمه به شرط هر برچسب را در یک فایل به نام ProbsB.txt ذخیره می‌کند.
Tags=**MyPOSTagger**(ProbsPi, ProbsA, ProbsB, TagSet, TestSet) این تابع احتمال‌های یک مدل HMM، لیست برچسب‌ها و فایل آزمون را گرفته، جملات تست را یکی یکی جدا کرده با فراخوانی تابع MyViterbi برای هر جمله، برچسب کلمات را بدست آورده و در نهایت جلو هر کلمه بنویسید.
Tags=**MyViterbi**(ProbsPi, ProbsA, ProbsB, TagSet, Sentence) این تابع احتمال‌های مدل HMM، لیست برچسب‌ها و یک جمله را گرفته و برچسب کلمات آن جمله را بدست می‌آورد.
Accuracy=**CalculatePOSTAccuracy**(TrueTags, EstimatedTags) این تابع دو فایل داده آزمون یکی حاوی برچسب‌های درست و دیگری حاوی برچسب‌های بدست آمده توسط الگوریتم برچسب‌زن را گرفته و با مقایسه برچسب‌های این دو، درصد برچسب‌های درست تشخیص داده شده را به عنوان خروجی برمی‌گرداند.

h.veisi@ut.ac.ir