فایل‌های ضمیمه : lm.ipynb شامل کدها و نتایج اجرا و lm.html خروجی کد و اجرا برای خوانایی بهتر

درباره ی داده ها :

فایل heartdisease.txt شامل متن تعدادی از صفحات وبلاگ فارسی درباره ی بیماری های قلبی است. این مجموعه به صورت دستی جمع آوری شده و شامل مطالبی از قبیل علایم انواع بیماری های قلبی، روش های درمان، نحوه ی عملکرد و ساختار قلب انسان، و مطالب مشابه است که در حوزه ی اطلاعات عمومی پزشکی درباره ی سلامت و بیماری قلب می‌باشد. این فایل فقط حاوی متن کپی شده و بهم چسبیده ی این صفحات است و هیچ گونه پردازشی روی آن انجام نشده. خط به خط فایل خوانده شده و روی text ذخیره شده.

قدم اول : پیش پردازش متن فارسی

با استفاده از کتابخانه ی parsivar پیش پردازش های زبان فارسی را انجام می‌دهم. اول هر خط text را نرمال می‌کنم. نرمال سازی متن کاراکترهایی که چند شکل نوشتاری فارسی یا فارسی-عربی دارند را به یک نوع کاراکتر استاندارد فارسی تبدیل می‌کند. فاصله ها با علایم نگارشی و فاصله و نیم فاصله ها را تنظیم می‌کند. متن نرمال شده در لیست text_norm قرار دارد. با parsivar متن را به صورت مجموعه ای از جملات واحدسازی می‌کنم. سپس با تابع دیگری از همان مدل Tokenizer هر جمله ی متن را به لیستی کلمات تشکیل دهنده اش واحدسازی می کنم. لیست tokenized_sentences مجموعه ی این لیست هاست. چون در توکن سازی علایم نگارشی هم هر کدام به عنوان یک توکن گرفته شدند و در مدل سازی زبانی که قرار است انجام دهم به علایم نگارشی نیازی ندارم، تمام توکن هایی که در tokenized_sentences موجود است و فقز یک علامت نگارشی هست را از لیست حذف می‌کنم. در کل ۲۶۹۶ لیست (جمله ی واحدسازی شده) دارم. برای اینکه دو مجموعه ی تست و آموزش داشته باشم یک نمونه گیری تصادفی از این لیست انجام می‌دهم و ۲۰۰ جمله را برمی‌دارم این مجموعه ی ارزیابی است و هر چه باقی می‌ماند مجموعه ی اموزش است. تعداد توکن های هر لیست را شمارش می‌کنم و به ترتیب ۲۴۵۸ و ۳۱۲۷۱ کلمه/توکن در هر مجموعه دارم.

قدم دوم : مدلسازی زبانی بدون هموارسازی و سپس افزودن توابع مختلف هموارساز

علی رغم توصیه به کار با ابرازهای نام برده شده در صورت تمرین از جمله SRILM به دلیل این که در فرایند نصب این ابزارها مشکلاتی داشتم و نتوانستم نحوه ی کار با آن ها را یادبگیرم ، برای مدلسازی از کتابخانه ی nltk استفاده کردم. لازم به تذکر است که بخش هایی از این کد با جدیدترین نسحه ی این nltk قابل اجرا هستند و از آنجا که این نسخه با نسخه ی فعلی parsivar دچار اشکلاتی هست پیشنهاد می‌کنم قبل از اجرای این بخش به بعد، موقتا parsivar را غیرفعال کنید و nltk را به نسخه ی جدید ارتقا دهید.

تابع n_gramModel در تمام بخش های کار به صورت تقریبا ثابتی حضور دارد و در تمام مراحل برای n-gram خواسته شده (unigram, bigram, trigram) توکن های <s> و <\s> را به ازای ابتدا و انتهای متن آموزشی اضافه می‌کند و بعد ماتریس جملات توکن سازی شده را به یک لیست یک بعدی تبدیل می‌کند. تمام ترکیب های n-gram را استخراج و شمارش می‌کند و در آخر با محاسبه ی احتمال ها، تعداد کلماتی که مدل با انها ساخته شده را چاپ می‌کند و مدل زبانی را باز می‌گرداند. چون با دیتاست کوچکی کار می‌کنم تعداد کلمات مدل زبانی را برابر تمام کلماتی گذاشتم که در متن آموزش بیش از یک بار دیده می‌شوند. به ازی کلمات غریبه مدل از <unk> استفاده می‌کند. تنها تفاوت این تابع در بخش های مختلف کار، استفاده از روش های مختلف هموارسازی است. تمام این مراحل با مدل ها و تابع کتابخانه ی nltk انجام گرفته. ضمنا برای اینکه مدل به‌دست آمده قابل اتکا تر باشه، به جای اینکه فقط n-gram ها شمارش شوند، m-gram ها به ازای m<=n شمرده شده اند. برای بخش اول n_gramModel با مدل MLE کار می‌کند که هیچ گونه هموارسازی انجام نمی‌دهد. در این بخش اول خارج از این تابع برای اینکه مجموعه ی تست را هم آماده کنم اول افزودن توکن های < و <\s> را به ازای

ابتدا و انتهای متن و سپس به لیست یک بعدی تبدیل کردم و برای هر سه مدل unigram, bigram, trigram توکن ها را استخراج کردم تا برای محاسبه ی انتروپی و سرگشتگی استفاده شوند.

در ادامه ی کار و در هر قدم از یک روش هموار سازی استفاده کردم که همین تابع n_gramModel را با تغیر مدل زبانی استفاده شده اجرا می‌کند و هر بار سرگشتگی و انتروپی را محاسبه می کند. در جدول زیر نام هر روش هموارسازی استفاده شده به ترتیب از روی فایل ضمیمه آمده و نتایج آن هم ذکر شده.

بدون هموار سازی		
unigram	bigram	trigram
model vocabulary = 3614 perplexity = inf entropy = inf	model vocabulary = 3616 perplexity = inf entropy = inf	model vocabulary = 3616 perplexity = inf entropy = inf
add-k یا Lidstone		
unigram	bigram	trigram
perplexity = 656.4 entropy = 9.4	perplexity = 1552.8 entropy = 10.6	perplexity = 2714.1 entropy = 11.4
add-one یا Laplace		
unigram	bigram	trigram
perplexity = 656.4 entropy = 9.4	perplexity = 1219.9 entropy = 10.3	perplexity = 2302.1 entropy = 11.2
KneserNey		
unigram	bigram	trigram
به دلیل خطای تقسیم بر صفر سرگشتگی و انتروپی قابل محاسبه نبود.	perplexity = inf entropy = inf	perplexity = inf entropy = inf
WittenBell		
unigram	bigram	trigram
perplexity = inf entropy = inf	perplexity = inf entropy = inf	perplexity = inf entropy = inf
Backoff (Stupid Backoff)		
unigram	bigram	trigram
perplexity = inf entropy = inf	perplexity = inf entropy = inf	perplexity = inf entropy = inf
AbsoluteDiscounting		
unigram	bigram	trigram
perplexity = inf entropy = inf	perplexity = inf entropy = inf	perplexity = inf entropy = inf

به غیر از دو حالت add-one و add-k (kرا برابر ۰.۰۱ گرفتم) در تمام حالت های دیگر مدل مقدار بی نهایت را برای عدد انتروپی و سرگشتگی به ازای تمام unigram, bigram, trigram ها نتیجه دادند. این برای خالت بدون هموارسازی نتیجه ی قابل تصوری هست اما در حالت هایی که از روش های هموارسازی هم استفاده شده انتظار چنین نتیجه را نداشتم. تصور می‌کنم چون با مدلسازی آشنا نیستم و برای بار اول هم هست با nltk کار می‌کنم شاید در استفاده از توابع و مدلها دچار خطا شده باشم و در واقع تمام این نتیجه ها به عیر از آن دو مورد همان نتیجه های بدون هموارسازی باشند! اما متوجه نشدم که این خطا از کجاست یا اینکه این تصور اصلا درست است یا نه. از طرف

دیگر در صورتی که کلمات متن تست برای مدل نااشنا باشند یا اینکه تعداد کلماتی که با انها متن قابل مدلسازی است نسبت به سایز مجموعه ی آموزش بزرگ باشد، سرگشتگی مدل بسیار زیاد و inf خواهد بود چون کلماتی که داخل تست هستند درون مدل همه ناشناخته خواهند بود.

بنابراین تمام تحلیلی که از این مدلسازی می‌توانم ارایه دهم مربوط به مقایسه ی دو حالت هموارسازی های add-one و add-k هستند که در کلاس هم مطرح شدند و تفاوتشان با خالت بدون هموارسازی. هموارسازی اساسا برای کاهش صفرهای ماتریس شمارش n-gram ها و احتمالهایشان استفاده می‌شود پس وقتی با هر دو روش نتیجه از حالت اول بهتر شد کاملا قابل انتطار یود. در هر دو روش لاپلاس و لیدستون به شمارش ها مقداری اضافه می‌شود تا آن مقدار صفر های گفته شده ی ماتریس های احتمال را از بین ببرد. در لاپلاس این مقدار اضافه شده ۱ و در لیدستون برای بهبود روش لاپلاس (مشکل تولید عددهای بزرگ) این مقدار اضافه شده از یک کمتر است که اینجا ۰,۰۱ گرفتم. هر دو روی unigram مقدار های ارزشیابی برابر می دهند اما وقتی با درجات بالاتر n-gram مدلسازی کردم تفاوت اشکار شد و این تفاوت هم در مقیاس لگاریتم (انتروپی) و هم به صورت قابل توجه تری روی خود سرگشتگی اشکارند که با trigram و bigram استفاده از Laplace از add-k, k=0.01 نتیجه ی بهتری دارد و روی این متن سرگشتگی کمتری نتیجه می‌دهد.

در تمام این ۶ مورد گزارش شده سزگشتگی و به طبع آن انتروپی با n های بیشتر، بیشتر می‌شوند. بهترین مدلی که در این کار به دست آوردم همان unigram با Laplace هست.