محمدرضا غفرانی
۴۰۰۱۳۱۰۷۶
۲۶ فروردین ۱۴۰۱

پردازش زبان طبیعی

تمرین اول

بخش صفر: جداسازی توکن‌ها

برای استخراج توکن‌ها در ابتدا می‌خواستیم از کتابخانه‌های معتبر نظیر هضم استفاده کنیم، اما با توجه به آن که توکن‌های استخراج شده توسط این کتابخانه تفاوتی با استخراج توکن‌ها با استفاده از کاراکتر فاصله نداشت، بنابراین ما کلمات را با استفاده از کاراکتر فاصله از هم جدا کردیم.

بخش اول: مدل‌های زبانی آماری

مدل زبانی Unigram

مدل زبانی Unigram به محتوای متن اهمیت نمی‌دهد و احتمال وقوع هر توکن را به تنهایی محاسبه می‌کند. این مدل در هنگام پیش‌بینی کلمات جا افتاده کلماتی را پیشنهاد می‌دهد که دارای بیشترین تعداد تکرار هستند. از آن جا که کلمات با بیشترین تعداد تکرار لزوما معنای جمله را تکمیل نمی‌کنند، بنابراین قدرت پیش‌بینی این مدل زبانی بسیار ضعیف است.

در مدل زبانی Unigram احتمال وقوع هر توکن بدون در نظر گرفتن Absolute Discounting به شکل زیر محاسبه می‌شود. در این فرمول منظور از $\#(w)$ تعداد رخداد توکن w و منظور از W مجموع تعداد رخداد تمامی توکن‌هاست.

$$P(w) = \frac{\#(w)}{W}$$

اگر ایده Absolute Discounting را به فرمول بالا اعمال کنیم فرمول محاسبه احتمال رخداد توکن به شکل زیر تغییر می‌کند. در این فرمول منظور از δ یک پارامتر ثابت است؛ پارامتر α نیز طبق فرمولی که در ادامه آورده می‌شود از روی δ قابل محاسبه است.

$$P(w) = \frac{\max(\#(w) - \delta, 0)}{W} + \alpha \times \frac{1}{V}$$

$$\alpha = \frac{\delta}{W} \times V$$

فرمول بالا تعمیمی است که من بر اساس اسلایدهای درس برای بهره‌گیری از ایده Absolute Discounting برای مدل Unigram انجام داده‌ام. در ایده Absolute Discounting از تمامی احتمال‌های مثبت مقدار اندکی برداشته شده و بدین ترتیب یک احتمال بزرگی ذخیره می‌شود. در ادامه این احتمال ذخیره‌شده به نسبت P_{BG} بین رخدادها تقسیم می‌شود. در این‌جا نیز از تمامی احتمال‌های مثبت که تعداد آن‌ها برابر اندازه کلمات(V) است، مقداری برداشته شده و سپس به نسبت P_{BG} که در این‌جا مدل zerogram است بین کلمات تقسیم می‌شود.

فرمول بالا با انجام عمل ساده‌سازی به شکل زیر تغییر می‌کند.

$$P(w) = \begin{cases} \frac{\#(w)}{W} & w \in Vocab \\ \frac{\delta}{W} & w \notin Vocab \end{cases}$$

همان‌طور که مشاهده می‌شود، این فرمول در واقع برای کلماتی که در مجموعه واژگان هستند، همان احتمال unigram را خروجی می‌دهد، اما برای کلماتی که در دایره واژگان نیستند یک احتمال بسیار کوچک را خروجی می‌دهد. با جست‌وجو در اینترنت برای یافتن تعمیم بهتر، به نتیجه خاصی نرسیدم. تنها نتیجه قابل بیان این لینک بود که توضیح می‌داد در هنگام اعمال ایده Kneser-Ney برای مدل Unigram نیز نتیجه مشابهی اتفاق می‌افتد. این نتیجه از آن‌جا قابل اهمیت است که ایده Kneser-Ney بسیار شبیه Absolute Discounting است.

برای محاسبه perplexity از فرمول زیر که در کتاب مرجع معرفی شده است، استفاده کرده‌ام.

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 w_3 ... w_N)}}$$

برای آن که پیاده‌سازی این فرمول مشکلات پیاده‌سازی نداشته باشد، از عبارت لگاریتم گرفته و مقدار آن را محاسبه می‌کنیم. البته در نهایت برای گزارش مقدار perplexity مجددا حاصل را به توان e می‌رسانیم. فرمول به شکل زیر در می‌آید.

$$\ln(PP(W)) = -\frac{1}{N} \ln(P(w_1 w_2 w_3 ... w_N))$$

برای مدل Unigram خواهیم داشت.

$$\ln(PP(W)) = -\frac{1}{N} \sum_{w_i} \ln(P(w_i))$$

در نهایت با ساخت مدل Unigram بر روی مجموعه متوجه می‌شویم که بهترین مقدار برای δ برابر 0.96 است. برای یافتن بهترین مقدار برای پارامتر δ، من بازه $(0, 1)$ را با اندازه قدم 0.1 بررسی کردم. همچنین به نظر می‌رسد که هر چقدر مقدار پارامتر δ بیشتر شود،

مقدار معیار perplexity کم‌تر می‌شود. کما این که در حال حاضر بزرگترین عدد قابل انتخاب برای پارامتر δ انتخاب شده است. بنابراین این سوال به نظر می‌رسد که چرا اعداد بزرگ‌تر از 1 بررسی نشده‌اند؟ در جواب این سوال گفت که هنگامی که اعداد بزرگ‌تر از یک را برای δ بررسی اعمال کنیم حاصل جمع توزیع احتمال بیشتر از یک می‌شود، که قابل قبول نیست.

با انتخاب پارامتر $\delta = 0.96$ مقدار perplexity بر روی مجموعه داده ارزیابی و آزمون به ترتیب برابر 1828.50 و 1810.07 می‌شود.

این مدل تمامی جاهای خالی موجود در فایل text_incomplete.txt را با توکن «و» پر می‌کند. این رفتار مدل قابل پیش‌بینی بود، چرا که همان‌طور که پیش‌تر ذکر شد این کلمه به محتوای متن اهمیت نداده و هر بار محتمل‌ترین توکن (که همان توکن «و» است) را پیش‌بینی می‌کند. در ادامه پیش‌بینی‌هایی که این مدل انجام داده است به همراه برچسب‌های مورد انتظار آورده شده است.

۱. این سخن حقست اگر نزد سخن گستر و (برند)
۲. آنکه با یوسف صدیق چنین خواهد و (کرد)
۳. هیچ دانی چکند صحبت او با و (دگران)
۴. سرمه دهی بصر بری سخت خوش است و (تاجری)
۵. آتش ابراهیم را و (نبود) و (زیان)
۶. من که اندر سر و (جنونی) و (داشتم)
۷. هر شیر شرزه را که به نیش و (سنان) و (گزید)
۸. هرکه از حق به و (سوی) و (او) و (نظریست)
۹. گفت این از و (خدای) و (باید) و (خواست)
۱۰. کلاه لاله که لعل است و و (تو) و (بشناسی)

مدل زبانی Bigram

مدل زبانی Bigram از آن جا که به محتوای متن اهمیت می‌دهد؛ بنابراین نسبت به مدل زبانی Unigram از قدرت بیشتری برخوردار است. در هنگام پیش‌بینی توکن‌های جدید این مدل به توکن قبلی اهمیت داده و به دلیل همین توکن‌های با معناتری را پیشنهاد می‌دهد.

احتمال رخداد هر توکن در مدل زبانی Bigram به صورت زیر محاسبه می‌شود. عملکرد $\#$ به همان ترتیب تعریف شده در بخش Unigram تعریف می‌شود.

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})}$$

با اعمال تکنیک Absolute Discounting فرمول بالا به شکل زیر تغییر می‌کند.

$$P(w) = \frac{\max(\#(w_{i-1}, w_i)) - \delta_1, 0)}{\#(w_{i-1})} + \alpha_1(\frac{\max(\#(w) - \delta_2, 0)}{W} + \alpha_2 \times \frac{1}{V})$$

$$\alpha_1 = \frac{\delta_1}{\#(w_{i-1})} \times \#(\#(w_{i-1}, w_i) > 0) \qquad \alpha_2 = \frac{\delta_2}{W} \times V$$

فرمول بالا در هنگام استفاده بر روی داده‌های تست به دلیل وجود توکن‌هایی که در مجموعه دادگان آموزشی وجود ندارد، به مشکل می‌خورد. علت اصلی این مشکل عبارت w_{i-1} است. از آن جا که در فرمول مشکل‌سازترین عبارت $\frac{\max(\#(w_{i-1}, w_i)) - \delta_1, 0)}{\#(w_{i-1})}$ است، از آن صرف‌نظر کرده و فرمول را برای محاسبه این کلمات به صورت زیر تعریف می‌کنیم.

$$P(w) = \frac{\max(\#(w) - \delta_2, 0)}{W} + \alpha_2 \times \frac{1}{V}$$

$$\alpha_2 = \frac{\delta_2}{W} \times V$$

در فرمول بالا منظور از V تعداد توکن‌های یکتای موجود در داده آموزشی، W تعداد کل توکن‌های موجود در دادگان آموزشی و منظور از عملگر $\#$ عملگر شمارشگر است برای مثال منظور از عبارت $\#(w_{i-1}, w_i)$ تعداد رخداد عبارت $w_{i-1}w_i$ است. این فرمول در واقع همان فرمول unigram است. بنابراین اگر کلمه w_{i-1} در دادگان آموزشی نباشد، ما احتمال unigram را برای کلمه w_i محاسبه می‌کنیم.

همچنین برای یافتن بهترین مقدار برای δ_1 و δ_2 اعداد موجود در بازه $(0, 1)$ را به ازای هر یک از δ_1 و δ_2 با اندازه قدم 0.1 بررسی کردیم. در نهایت مقادیر $\delta_1 = 0.9$ و $\delta_2 = 0.9$ به عنوان بهترین مقادیر گزارش شدند. با اجرای مدل با استفاده از این پارامترها perplexity مدل بر روی مجموعه داده ارزیابی برابر 1223.15 و برای مجموعه داده آزمون برابر 1217.71 می‌شود.

در نهایت مدل عبارات زیر را برای جاهای خالی موجود در داده‌های تست پیشنهاد می‌دهد. این مدل نیز توکن‌های مناسبی را برای کلمات پیشنهاد نمی‌دهد اما نسبت به مدل قبلی پیشنهادها بهتر است. مدل قبلی تمامی جاهای خالی را با توکن «و» پر کرد در حالی که این مدل توکن‌های مختلفی را برای جاهای مختلف پیشنهاد می‌دهد. دلیل این اتفاق اهمیت دادن این مدل به محتوای متن است.

۱. این سخن حقست اگر نزد سخن گستر و (برند)

۲. آنکه با یوسف صدیق چنین خواهد ر (کرد)

۳. هیچ دانی چکند صحبت او با و (دگران)

۴. سرمه دهی بصر بری سخت خوش است را (تاجری)

۵. آتش ابراهیم را و (نبود) از (زیان)

۶. من که اندر سر و (جنونی) از (داشتم)

۷. هر شیر شرزه را که به نیش در (سنان) آن (گزید)

۸. هرکه از حق به ر (سوی) و (او) از (نظریست)

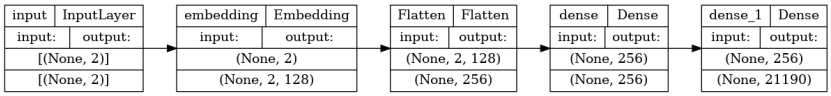
۹. گفت این از تو (خدای) در (باید) آن (خواست)

۱۰. کلاه لاله که لعل است را (اگر) به (تو) دست (بشناسی)

بخش دوم: مدل‌های عصبی

شبکه عصبی جلورو

در این قسمت می‌خواهیم با استفاده از شبکه‌های عصبی جلورو مدل زبانی را بسازیم. برای این کار از ابزار Tensorflow استفاده می‌کنیم. با استفاده از این ابزار شبکه عصبی جلوروی زیر را طراحی می‌کنیم. (شکل ۱) این شبکه عصبی از یک لایه ورودی، یک لایه پنهان با تعداد ۶۴ نورون و لایه خروجی تشکیل شده است.

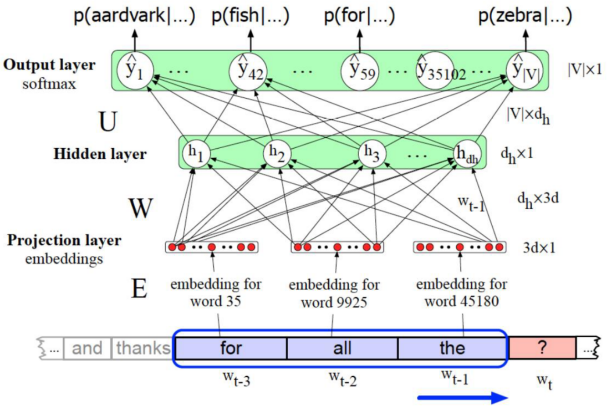


شکل ۱: ساختار شبکه‌ی عصبی جلوروی استفاده شده

برای کدگذاری توکن‌ها از تکنیک integer encoding استفاده کرده‌ام. بدین صورت که ابتدا لیست توکن‌های موجود در داده‌های آموزشی را یافته و سپس به هر یک از این توکن‌ها یک شناسه نسبت می‌دهیم. حال هنگامی که می‌خواهیم چند کلمه را به شبکه بدهیم، شناسه آن‌ها را از روی دیکشنری یافته کرده و سپس این شناسه‌ها را در اختیار شبکه قرار می‌دهیم. علت در نظر گرفتن این کدگذاری مصرف بسیار کم حافظه است، برای مثال اگر می‌خواستیم از روش one hot استفاده کنیم باید برای هر کلمه یک بردار به طول تعداد توکن‌ها در نظر می‌گرفتیم که به وضوح میزان مصرف بیشتری داشت، بعلاوه استفاده از کدگذاری one hot طبق بررسی‌هایی که انجام دادم تاثیر چندانی در عملکرد نهایی مدل نداشت. در قدم بعدی این اعداد به لایه Embedding داده می‌شود تا به هر کدام یک بردار نسبت داده شود.

از آن جا که در این شبکه نیز ممکن است به مشکل عدم وجود یک توکن در داده‌های آموزشی بخوریم، من به دیکشنری که از روی داده‌های آموزش ساخته می‌شود، یک کلمه به عنوان کلمه «ناآشنا» اضافه کرده‌ایم.

برای داده‌های خروجی از کدگذاری one-hot encoding بهره می‌گیریم. علت انتخاب این کدگذاری، معماری آورده شده برای شبکه‌های جلورو در اسلایدهای درس است. این شکل مجددا در شکل ۲ آورده می‌شود. همان‌طور که مشاهده می‌شود، خروجی شبکه یک لایه softmax دارد که احتمال رخداد هر توکن را به ازای مشاهده توکن‌های قبلی می‌دهد. از آن جا که در کدگذاری one hot برای بردار خروجی به ازای توکن خروجی عدد یک و برای بقیه توکن‌ها عدد صفر در نظر گرفته می‌شود، بنابراین مدل بهتر می‌تواند احتمال‌های خروجی را یاد بگیرد.



شکل ۲: ساختار شبکه‌ی عصبی جلورو موجود در اسلایدهای درس

همچنین ما از بهینه‌ساز Adam با نرخ یادگیری ثابت 0.1 و تابع خطای categorical crossentropy استفاده کرده‌ایم. نتایجی که در ادامه آورده می‌شود، با آموزش شبکه عصبی بر روی ۳۰،۰۰۰ سطر اول از داده‌های آموزشی حاصل شده است. با افزایش تعداد داده‌های آموزشی نتیجه حاصل شده مشابه حالت‌های گزارش شده است.

در حالت Bigram، با آموزش شبکه در طی ۱۰ گام مدل به همگرایی می‌رسد، اما نتایجی که شبکه به آن می‌رسد تعریفی ندارد. مدل در طی این آموزش به صحت 13 درصد و خطای 5.80 در داده‌های آموزشی و در داده‌های ارزیابی به صحت 7 درصد و خطای 8.16 دست پیدا می‌کند. در هنگامی که شبکه عصبی بر روی داده‌های حاصل از Trigram نیز آموزش می‌بیند، در داده‌های آموزشی به صحت 12.12 درصد و خطای 5.90 و در داده‌های ارزیابی به صحت 6.7 درصد و خطای 8.32 می‌رسد. دقت داشته باشید که بر اساس صورت تمرین، در مدل شبکه عصبی در حالت Bigram به جای یک کلمه، دو کلمه به شبکه داده شده و کلمه بعدی پیش‌بینی می‌شود. در حالت Trigram نیز به جای دو کلمه، سه کلمه به مدل داده شده و مدل کلمه بعدی را پیش‌بینی می‌کند. در حالت Trigram نیز تقریبا به ازای هر ورودی مدل کلمه «و» را به عنوان خروجی بعدی پیش‌بینی می‌کند.

بهتر شدن نتایج از مدل Bigram به مدل Trigram را با وجود پیچیده‌تر بودن مدل Trigram می‌توان این گونه توجیه کرد که تعداد داده‌های آموزشی کم‌تری را برای Trigram

در مقایسه با Bigram در نظر می‌گیریم. برای توضیح علت آن از مثال کمک می‌گیریم. عبارت «باز رویاند گل صباغ را» را در نظر بگیرید. اگر بخواهیم این عبارات را به Bigramهایش تقسیم کنیم خواهیم داشت: «باز رویاند گل»، «رویاند گل صباغ»، «گل صباغ را». این در حالی است که اگر این عبارت را به Trigramهایش تقسیم کنیم، عبارت‌های زیر حاصل می‌شود: «باز رویاند گل صباغ»، «رویاند گل صباغ را». همان‌طور که مشاهده می‌شود تعداد عبارات حاصل شده در حالت Trigram از Bigram کم‌تر است. از آن جا که هر دو مدل نیز عملا پیش‌بینی توکن «و» را یاد می‌گیرند بنابراین به دلیل کم‌بودن داده ارزیابی، مدل Trigram راحت‌تر می‌تواند به نتایج بهتر دست یابد.

در ادامه پیش‌بینی که مدل شبکه عصبی با داده‌های حاصل از Bigram آموزش دیده است، آورده شده است. همان‌طور که مشاهده می‌شود این مدل شبکه عصبی نتوانسته است، جاهای خالی را به خوبی پر بکند.

۱. این سخن حقست اگر نزد سخن گستر از (برند)
۲. آنکه با یوسف صدیق چنین خواهد از (کرد)
۳. هیچ دانی چکند صحبت او با در (دگران)
۴. سرمه دهی بصر بری سخت خوش است و (تاجری)
۵. آتش ابراهیم را و (نبود) به (زیان)
۶. من که اندر سر آن (جنونی) به (داشتم)
۷. هر شیر شرزه را که به نیش جان (سنان) را (گزید)
۸. هرکه از حق به بر (سوی) سر (او) سر (گزید)
۹. گفت این از معنی (خدای) آن (باید)که (خواست)
۱۰. کلاه لاله که لعل است و (اگر) و (تو) از (بشناسی)

عملکرد مدل شبکه عصبی هنگام آموزش بر روی داده‌های Trigram مشابه است. در این حالت نیز کلمات پیشنهادی، کلمات چندان جالبی برای پر کردن جاهای خالی نیستند.

۱. این سخن حقست اگر نزد سخن گستر تو (برند)
۲. آنکه با یوسف صدیق چنین خواهد را (کرد)
۳. هیچ دانی چکند صحبت او با نه (دگران)
۴. سرمه دهی بصر بری سخت خوش است و (تاجری)
۵. آتش ابراهیم را و (نبود) و (زیان)

۶. من که اندر سر او (جنونی) جهان (داشتم)

۷. هر شیر شرزه را که به نیش بر (سنان) دست (گزید)

۸. هرکه از حق به تو (سوی) و (او) دست (گزید)

۹. گفت این از و (خدای) و (باید)آن (خواست)

۱۰. کلاه لاله که لعل است در (اگر) و (تو) و (بشناسی)

در نهایت عملکرد شبکه‌های عصبی را از نظر perplexity گزارش می‌کنیم. perplexity شبکه عصبی آموزش دیده با داده‌های bigram در داده‌های آزمون برابر 3522.02 و عملکرد مدل آموزش دیده با داده‌های Trigram برابر 4129.80 می‌شود. همان‌طور که مشاهده می‌شود شبکه‌های عصبی بر خلاف انتظار از عملکرد ضعیف‌تری برخوردار هستند. دلایل مختلفی را می‌توان برای توجیه این اتفاق آورد.

- کم بودن تعداد داده‌های آموزشی برای شبکه‌های عصبی. همان‌طور که می‌دانید شبکه‌های عصبی قدرت خود را در داده‌های حجیم نشان می‌دهند. در حالی که ما برای شبکه‌های عصبی به دلیل در نظر گرفتن ngramای بزرگ‌تر ناخواسته تعداد داده‌های کم‌تری را فراهم کرده‌ایم.
- عدم استخراج درست توکن‌ها. همان‌طور که پیش‌تر توضیح داده شد، ما برای استخراج توکن‌ها از کاراکتر فاصله بین کلمات استفاده کرده‌ایم. به دلیل ماهیت شعر و تمیز نبودن داده‌ها گاهی توکن‌های نامناسب استخراج شده است. برای مثال در مجموعه داده بیت «یکی د ر ملک شاهنشه یکی چون کوشش حیدر» وجود دارد، همان طور که مشاهده می‌شود کلمه «در» به اشتباه به صورت «د ر» نوشته شده است. این اشتباه باعث شده است که این کلمه به صورت اشتباه استخراج شود.

البته شبکه‌های عصبی همچنان مزیت‌هایی نیز دارند، برای مثال در هنگام پر کردن جاهای خالی در اشعار با استفاده از شبکه‌های عصبی می‌توان با دادن چند توکن قبلی، توکن بعدی را پیش‌بینی کرد در حالی که در مدل‌های آماری نظیر bigram احتمال رخداد تمامی کلمات موجود در V را محاسبه کرده و بهترین کلمه را برداشت.
جدای از عملکرد ضعیف شبکه‌های عصبی نسبت به مدل‌های آماری، عملکرد کلی مدل‌ها ضعیف است. این موضوع به وضوح در هنگام پیش‌بینی توکن‌های خالی قابل مشاهده است. بیشتر توکن‌های پیشنهاد شده توسط این شبکه‌ها حروف اضافه و کلمات پرکاربردی نظیر «تو» است که مفهوم چندانی را منتقل نمی‌کنند. بعلاوه صحت شبکه عصبی از لحاظ کمی پایین است. اگر شبکه عصبی به ازای تمامی مقادیر ورودی کلمه «و» را پیش‌بینی کند می‌تواند به دقت ۵ درصد برسد، به همین دلیل رسیدن به دقت ۶ یا ۷ درصد در مجموعه داده ارزیابی عملکرد چندان خوبی نیست.