به نام خدا

تکلیف سری چهارم NLP

بهاره کاوسی نژاد – 99431217

سوالات تئوری

1. دو جدول زیر تعداد Unigram ها و Bigram ها در یک پیکره فرضی را نشان می دهد.

ما	خواندیم	دیروز	امروز	داستان	کتاب
۱۸۷۲	۱۴۹۵	۲۰۲۱	۱۹۴۳	۹۴۵	۱۲۴۵

	ما	خواندیم	دیروز	امروز	داستان	کتاب
ما	۰	۱۵۶	۴۱۱	۴۵۲	۲۳۸	۳۸۷
خواندیم	۲	۰	۶	۱۱	۸۴	۱۱۲
دیروز	۳۴۱	۳۲	۰	۴۸	۶۸	۲۵۴
امروز	۳۲۸	۱۲	۰	۰	۸۷	۲۳۱
داستان	۴	۳۴۵	۸۴	۳۸	۰	۱
کتاب	۳۱	۳۲۰	۳	۰	۴۰۳	۰

احتمال رخداد جملات تست زیر را محاسبه کنید. فرض بر این است که جملات تست در وسط یک رشته هستند. یعنی در نظر گرفتن احتمال بندهای شروع و پایان جمله الزام نیست.

جمله تست ۱ : ... ما امروز کتاب خواندیم...

1. ابتدا احتمال رخداد هر کلمه را با توجه به unigram محاسبه می کنیم:

- P(ما) = Count(ما) / Total Count of Unigrams = 1872 / (1872 + 1495 + 2021 + 1943 + 945 + 1245) = 1872 / 9521 = 0.196
- P(امروز) = Count(امروز) / Total Count of Unigrams = 1943 / 9521 = 0.204
- P(کتاب) = Count(کتاب) / Total Count of Unigrams = 1245 / 9521 = 0.130
- P(خواندیم) = Count(خواندیم) / Total Count of Unigrams = 1495 / 9521 = 0.157

2. احتمال رخداد هر bigram را با توجه به جدول bigramها محاسبه می کنیم:

- P(ما|امروز) = Count(ما امروز) / Count(ما) = 452 / 1872 = 0.241
- P(امروز|کتاب) = Count(امروز کتاب) / Count(امروز) = 231 / 1943 = 0.118
- P(کتاب|خواندیم) = Count(کتاب خواندیم) / Count(کتاب) = 320 / 1245 = 0.257

3. احتمال نهایی را با ضرب احتمالات کلمات یا bigramها بدست می آوریم:

- P(ما امروز کتاب خواندیم) = P(ما) * P(امروز|ما) * P(کتاب|امروز) * P(خواندیم|کتاب) = 0.196 * 0.241 * 0.118 * 0.257 = 0.001

جمله تست ۲ : ... ما دیروز داستان خواندیم ...

- ابتدا احتمال رخداد هر کلمه را با توجه به unigram محاسبه می کنیم:

- P(ما) = Count(ما) / Total Count of Unigrams = 1872 / (1872 + 1495 + 2021 + 1943 + 945 + 1245) = 1872 / 9521 = 0.196
- P(دیروز) = Count(دیروز) / Total Count of Unigrams = 2021 / 9521 = 0.212
- P(داستان) = Count(داستان) / Total Count of Unigrams = 945 / 9521 = 0.099
- P(خواندیم) = Count(خواندیم) / Total Count of Unigrams = 1495 / 9521 = 0.157

- احتمال رخداد هر bigram را با توجه به جدول bigramها محاسبه می کنیم:

- P(دیروز|ما) = Count(ما دیروز) / Count(ما) = 411 / 1872 = 0.219
- P(داستان|دیروز) = Count(دیروز داستان) / Count(دیروز) = 68 / 2021 = 0.033
- P(خواندیم|داستان) = Count(داستان خواندیم) / Count(داستان) = 345 / 945 = 0.365

- احتمال نهایی را با ضرب احتمالات کلمات یا bigramها بدست می آوریم:

- P(ما دیروز داستان خواندیم) = P(ما) * P(دیروز|ما) * P(داستان|دیروز) * P(خواندیم|داستان) = 0.196 * 0.219 * 0.033 * 0.365 = 0.0005

2. رابطه زیر را اثبات کنید.

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\dots P(w_n|w_1^{n-1}) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

دنباله n کلمه را یا به صورت $w_1 \dots w_n$ و یا به صورت w_1^n نمایش می دهیم. بنابراین عبارت w_1^{n-1} به معنای رشته $w_1, w_2, \dots, w_{n-1}$ است. برای joint probability هر کلمه در یک دنباله که به این صورت

$P(X_1 = w_1, X_2 = w_2, X_3 = w_3, \dots, X_n = w_n)$

نوشته می شود از $P(w_1,w_2,\dots,w_n)$ استفاده می کنیم.

برای محاسبه احتمال کل دنباله مانند $P(w_1,w_2,\dots,w_n)$ می توانیم از chain rule of probability استفاده کنیم.

Chain rule — Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. Let $A_1, \ldots, A_n \in \mathcal{A}$. Then

$$\mathbb{P}(A_1 \cap A_2 \cap \ldots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2 \mid A_1)\mathbb{P}(A_3 \mid A_1 \cap A_2) \cdot \ldots \cdot \mathbb{P}(A_n \mid A_1 \cap \cdots \cap A$$
$$= \mathbb{P}(A_1) \prod_{j=2}^{n} \mathbb{P}(A_j \mid A_1 \cap \cdots \cap A_{j-1}).$$

Proof

The formula follows immediately by recursion

$$\begin{aligned} (1) \quad & \mathbb{P}(A_1)\mathbb{P}(A_2 \mid A_1) && = && \mathbb{P}(A_1 \cap A_2) \\ (2) \quad & \mathbb{P}(A_1)\mathbb{P}(A_2 \mid A_1)\mathbb{P}(A_3 \mid A_1 \cap A_2) && = && \mathbb{P}(A_1 \cap A_2)\mathbb{P}(A_3 \mid A_1 \cap A_2) \\ & && = && \mathbb{P}(A_1 \cap A_2 \cap A_3), \end{aligned}$$

where we used the definition of the conditional probability in the first step.

$$P(X1 \ldots Xn) = P(X1)P(X2|X1)P(X3|X1{:}\,2) \ldots P(Xn|X1{:}\,n-1) = \prod_{k=1}^{n} P(Xk\,|X1{:}\,k-1)$$

با اعمال chain rule به کلمات خواهیم داشت:

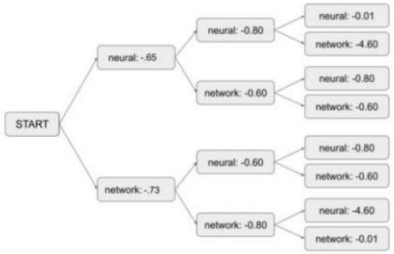
$$P(w1{:}\,n) = P(w1)P(w2|w1)P(w3|w1{:}\,2) \ldots P(wn|w1{:}\,n-1) = \prod_{k=1}^{n} P(wk\,|w1{:}\,k-1)$$

Chain rule ارتباط بین محاسبه joint probability یک دنباله را نشان می دهد و محاسبه احتمال شرطی یک کلمه با کلمات قبلی. معادله بالا نشان می دهد که ما می توانیم joint probability یک دنباله کامل کلمات را با ضرب تعدادی احتمال شرطی در یکدیگر تخمین بزنیم.

3. روش search beam را می توان به صورت شبه کد زیر نوشت:

Algorithm 1 Beam Search

for each time step t **do**
  for each hypothesis $y_{1:t-1,i}$ that we are tracking **do**
    find the top k tokens $y_{t,i,1}, \ldots, y_{t,i,k}$
  end for
  sort the resulting k^2 length t sequences by their total log-probability
  store the top k
  advance each hypothesis to time $t+1$
end for



Beam search را برای کدگشایی دنباله ای به طول 3 با k = 2 اجرا میکنیم. پیشبینی های یک کدگشا را در شکل بالا در نظر بگیرید، که در آن هر گره در درخت نشان دهنده پیشبینی احتمال ثبت توکن بعدی مشروط به توکن های قبلی در یک مرحله از کدگشا است (اعداد نشان داده شده، لگاریتم پیشبینی احتمال (log probability prediction) کدگشا از توکن فعلی با توجه به توکنهای قبلی است).

Vocabulary از دو کلمه تشکیل شده است:

"network", "neural"

a. در مرحله زمانی 1، beam search کدام دنباله را ذخیره می کند؟

تنها دو گزینه وجود دارد، پس beam search هر دو را نگه می دارد:

- "neural" (log prob = -.65)
- "network" (log prob = -.73)

b. در مرحله زمانی 2، beam search کدام دنباله را ذخیره می کند؟

تمام دنباله های دو کلمه ای را در نظر میگیریم، اما فقط می توانیم دوتای اول را نگه داریم:

- "neural network" (log prob = -.65 - .6 = -1.25)
- "network neural" (log prob = -.73 - .6 = -1.33)

c. در مرحله زمانی 3، beam search کدام دنباله را ذخیره می کند؟

دنباله های سه کلمه ای را در نظر میگیریم که با "neural network" و "network neural" آغاز می شوند و دوتای اولی به شکل زیر هستند:

- "neural network network" (log prob = -.65 - .6 - .6 = -1.85)
- "network neural network" (log prob = -.73 - .6 - .6= -1.93)

d. آیا beam search دنباله ی کلی با بیشترین احتمال (overall most-likely sequence) در این مثال را برمی گرداند؟ توضیح دهید.

خیر، دنباله کلی با بیشترین احتمال برابر با "neural neural neural" با احتمال زیر است:

log prob = -.65 - .8 - .01= -1.46

این دنباله ها بازگردانده نمی شوند زیرا در مرحله 2 حذف می شوند و در نظر گرفته نمی شوند (زیرا "neural neural" در دنباله ای با بیشترین احتمال و k=2 در دنباله های دو کلمه ای نیست).

e. پیچیدگی زمان اجرا تولید یک دنباله با طول T با اندازه پرتو k با RNN چقدر است؟ برحسب T و k و M پاسخ دهید (M اندازه vocabulary است).

- Step RNN forward one step for one hypothesis = O(M)

زیرا برای هر کلمه در لغات یک logit حساب می کنیم و عملیات های RNN دیگر بر M، T و K وابستگی ندارند.

- Do the above, and select the top k tokens for one hypothesis.

این کار را با sort کردن logitها انجام می دهیم؛ در نتیجه داریم: O(M logM)

راه هایی با پیچیدگی کمتر برای انتخاب K مورد اول وجود دارد (به عنوان مثال استفاده از min heap). از این روش استفاده می کنیم زیرا پیاده سازی آن آسان تر است.

در نتیجه با در نظر گرفتن مرحله قبلی تا کنون به این پیچیدگی رسیده ایم:

$O(M \log M + M) = O(M \log M)$.

- مراحل بالا را K بار برای current hypotheses انجام می دهیم و داریم: $O(KM \log M)$
- مراحل بالا + انتخاب K مورد اول از K^2 تا hypothesesی که اکنون ذخیره شده اند:

این کار را با sort کردن آرایه ای از K^2 مورد انجام می دهیم؛ یعنی: $O(K^2 \log(K^2))$. از آنجایی که $\log(K^2) = 2\log(K)$ می توان پیچیدگی را به این صورت نوشت: $O(K^2\log(K))$

(راه هایی با پیچیدگی کمتر نیز برای این مرحله وجود دارند)

با در نظر گرفتن مراحل قبل تر داریم: $O(K M \log M + K^2 \log (K))$

از آنجایی که M بزرگتر مساوی K است می توان نوشت: $O(K M \log M)$

این کار را برای T تا timestep انجام می دهیم: $O(T K M \log M)$

4. به سوالات زیر پاسخ دهید.

a. اگر در LSTM فقط بخواهیم گیت forget را داشته باشیم و گیت های input و output را حذف کنیم، چه اتفاقی میافتد و خروجی چه تغییری می کند؟

در یک LSTM استاندارد، گیت ورودی کنترل می کند که چه مقدار از اطلاعات جدید باید در سلول حافظه ذخیره شود، و دروازه خروجی تعیین می کند که چه مقدار از محتوای سلول باید خارج شود. این گیت ها نقش مهمی در تنظیم جریان اطلاعات داخل و خارج سلول حافظه دارند.

با حذف گیت های ورودی و خروجی، LSTMتوانایی کنترل و تنظیم جریان اطلاعات را از دست می دهد و در نتیجه عملکرد LSTM تغییر می کند. به طور خاص، بدون گیت ورودی، LSTM دیگر توانایی به روز رسانی selective و ذخیره اطلاعات جدید در سلول حافظه را نخواهد داشت. این می تواند منجر به مشکلاتی در یادگیری و به خاطر سپردن dependency های طولانی مدت در داده ها شود.

علاوه بر این، بدون گیت خروجی، LSTM قادر به کنترل مقدار و زمان بندی اطلاعات خروجی از سلول حافظه نخواهد بود. این می تواند بر توانایی LSTM برای تولید خروجی های دقیق و معنی دار تأثیر بگذارد.

به طور کلی، حذف گیت های ورودی و خروجی از یک LSTM به طور قابل توجهی عملکرد آن را تغییر می دهد و ممکن است منجر به کاهش عملکرد در کارهایی شود که به توانایی یادگیری و به خاطر سپردن dependency های طولانی مدت و تولید خروجی دقیق نیاز دارند.

b. اگر در یک LSTM مقدار گیت forget را به صفر تنظیم کنیم، چه اتفاقی می افتد و چطور این تغییر تاثیری بر روی توانایی شبکه در یادگیری و پیش بینی دارد؟

اگر مقدار گیت forget را در یک شبکه LSTM صفر کنیم، به این معنی است که LSTM حافظه قبلی خود را به طور کامل فراموش می کند و هیچ اطلاعاتی از مرحله زمانی قبلی را حفظ نمی کند. گیت forget در LSTM کنترل می

کند که چه مقدار از حافظه قبلی باید فراموش یا حفظ شود. با قرار دادن مقدار گیت forget روی صفر، ما به طور موثر به LSTM دستور می دهیم تا تمام اطلاعات قبلی را از سلول حافظه خود پاک کند. این پیامدهای قابل توجهی برای توانایی شبکه در یادگیری و پیش بینی دارد، زیرا قادر به حفظ وابستگی طولانی مدت یا به خاطر سپردن اطلاعات گذشته نخواهد بود. LSTM اساساً مانند یک شبکه عصبی feedforward استاندارد، بدون هیچ گونه حافظه یا توانایی برای حفظ متن عمل می کند.

برخی از اثرات خاص تنظیم مقدار گیت forget روی صفر:

1. **از دست دادن dependency های طولانی مدت:** LSTM نمی تواند dependency های طولانی مدت را در دنباله ورودی یاد بگیرد و به خاطر بسپارد. فقط مرحله زمانی فعلی را در نظر می گیرد و از ورودی های گذشته اطلاعی نخواهد داشت.
2. **ناتوانی در مدیریت اطلاعات sequential:** LSTM ها برای مدیریت داده های sequential و گرفتن وابستگی های زمانی طراحی شده اند. با این حال، با صفر کردن گیت forget، شبکه این قابلیت را از دست می دهد و قادر به پردازش و مدل سازی موثر اطلاعات sequential نیست.
3. **قدرت پیش بینی محدود:** بدون حفظ اطلاعات قبلی، قدرت پیش بینی LSTM به شدت محدود خواهد شد. نمی تواند پیش بینی های آگاهانه ای را بر اساس بافت یا الگوهای گذشته در داده ها انجام دهد.
4. **دشواری در یادگیری taskهای پیچیده:** کارهایی که نیاز به مدل سازی و درک dependency های طولانی مدت دارند، مانند ترجمه زبان یا تشخیص گفتار، به طور قابل توجهی برای LSTM چالش برانگیزتر خواهند شد. توانایی شبکه برای یادگیری روابط و الگوهای پیچیده در داده ها کاهش خواهد یافت.

به طور خلاصه، تنظیم مقدار گیت forget روی صفر اساساً جنبه حافظه LSTM را حذف می کند و در نتیجه توانایی شبکه برای یادگیری و پیش بینی بر اساس اطلاعات گذشته از بین می رود. این به طور قابل توجهی ظرفیت شبکه را برای رسیدگی به داده های sequantial و taskهای وابسته به زمینه محدود می کند.

c. توضیح دهید که چگونه افزایش تعداد لایه های LSTM در یک شبکه می تواند به کارایی و عملکرد شبکه کمک کند یا باعث افزایش پیچیدگی شود.

افزایش تعداد لایه های LSTM در یک شبکه می تواند اثرات مثبت و منفی بر کارایی، عملکرد و پیچیدگی شبکه داشته باشد:

1. **کارایی (efficiency):** افزودن لایه های LSTM بیشتر به طور بالقوه می تواند کارایی شبکه را در موارد خاص بهبود بخشد. با معماری های عمیق تر، شبکه می تواند الگوها و وابستگی های پیچیده تری را در داده ها ثبت کند که به طور بالقوه منجر به عملکرد بهتر با پارامترهای کمتر می شود. این به این معنی است که یک شبکه با چندین لایه LSTM ممکن است در مقایسه با یک شبکه کم عمق با تعداد پارامترهای یکسان، دقت مشابه یا بهتری را به دست آورد که در نتیجه کارایی بهبود یافته است.

2. **عملکرد (performance):** افزایش تعداد لایه های LSTM می تواند توانایی شبکه را برای یادگیری و مدل سازی الگوهای پیچیده در داده ها افزایش دهد. معماری های عمیق تر امکان استخراج ویژگی های سلسله مراتبی و نمایش مفاهیم انتزاعی تر را فراهم می کنند. این می تواند منجر به بهبود عملکرد در کارهایی شود که نیاز به گرفتن وابستگی های طولانی مدت یا مدیریت ساختارهای پیچیده دارند. با این حال، افزایش عملکرد ممکن است همیشه قابل توجه نباشد و می تواند به دامنه مسئله و مجموعه داده خاص بستگی داشته باشد.
3. **پیچیدگی (complexity):** افزودن لایه های LSTM بیشتر، پیچیدگی شبکه را افزایش می دهد. معماری‌های عمیق‌تر پارامترهای بیشتری را معرفی می‌کنند و ظرفیت مدل را برای یادگیری و نمایش روابط پیچیده افزایش می‌دهند. با این حال، این به قیمت افزایش پیچیدگی محاسباتی و نیازهای حافظه نیز تمام می شود. آموزش شبکه های عمیق تر می تواند چالش برانگیزتر باشد و ممکن است به داده ها و منابع محاسباتی بیشتری نیاز داشته باشد.
4. **Overfitting:** معماری های عمیق تر LSTM خطر بیشتری برای overfitting دارند، به خصوص زمانی که مجموعه داده کوچک باشد. شبکه به طور بالقوه می تواند داده های آموزشی را خیلی خوب به خاطر بسپارد و در نتیجه تعمیم ضعیفی به نمونه های دیده نشده دارد. تکنیک‌هایی مانند regularization، dropout یا early stopping را می‌توان برای کاهش overfitting در شبکه‌های عمیق‌تر به کار برد.
5. **دشواری آموزش:** آموزش شبکه های LSTM عمیق تر می تواند چالش برانگیزتر از معماری های کم عمق باشد. گرادیان های vanishing یا exploding مسائل رایجی هستند که می توانند در شبکه های عمیق ایجاد شوند. تکنیک هایی مانند gradient clipping، batch normalization یا استفاده از residual connections می توانند به تثبیت روند آموزش در معماری های عمیق تر کمک کنند.
6. **Interpretability:** به دلیل افزایش تعداد لایه‌ها، interpret یا درک شبکه‌های LSTM عمیق‌تر ممکن است دشوارتر شود. جریان اطلاعات و استخراج ویژگی‌ها انتزاعی‌تر و پیچیده‌تر می‌شوند و تشخیص اینکه کدام ویژگی‌ها برای پیش‌بینی‌ها مهم هستند، دشوارتر می‌شوند.

به طور خلاصه، افزایش تعداد لایه های LSTM در یک شبکه می تواند معاوضه هایی داشته باشد. به طور بالقوه می تواند کارایی و عملکرد شبکه را با اجازه دادن به شبکه برای گرفتن الگوها و وابستگی های پیچیده تر بهبود بخشد. با این حال، این ممکن است پیچیدگی محاسباتی، دشواری آموزش و خطر overfitting را نیز افزایش دهد.

منابع:

- https://en.wikipedia.org/wiki/Chain_rule_(probability)
- https://inst.eecs.berkeley.edu/~cs182/sp23/assets/section/dis07/sol07.pdf
- https://www.pluralsight.com/resources/blog/guides/introduction-to-lstm-units-in-rnn