بخش تئوری

سوال اول

برای محاسبهٔ احتمال جملات داده شده، با استفاده از احتمالات Bigramهای موجود در آن، از رابطهٔ زیر استفاده می‌کنیم:

$$p(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)}$$

همچنین برای محاسبهٔ احتمال کل جمله، بر اساس رابطهٔ زیرین که بر اساس Chain Rule و Markov Chain به دست آمده است، پیش می‌رویم:

$$p(w_1, w_2, \dots, w_n) = p(w_n|w_{n-1})p(w_{n-1}|w_{n-2}) \dots p(w_2|w_1)$$

جملهٔ تست ۱:

داریم:

$$p\left(\text{امروز}\middle|\text{ما}\right) = p(w_2|w_1) = \frac{452}{1872}$$
$$p\left(\text{کتاب}\middle|\text{امروز}\right) = p(w_3|w_2) = \frac{231}{1943}$$
$$p\left(\text{خواندیم}\middle|\text{کتاب}\right) = p(w_4|w_3) = \frac{320}{1245}$$

بنابراین داریم:

$$p(w_1, w_2, \dots, w_n) = p(w_n|w_{n-1})p(w_{n-1}|w_{n-2}) \dots p(w_2|w_1)$$
$$p(w_1, w_2, \dots, w_n) = \frac{452}{1872} \times \frac{231}{1943} \times \frac{320}{1245}$$
$$\rightarrow p(w_1, w_2, \dots, w_n) \approx 0.0073$$

جملهٔ تست ۲:

داریم:

$$p\left(\text{دیروز}\middle|\text{ما}\right) = p(w_2|w_1) = \frac{411}{1872}$$
$$p\left(\text{داستان}\middle|\text{دیروز}\right) = p(w_3|w_2) = \frac{68}{2021}$$
$$p\left(\text{خواندیم}\middle|\text{داستان}\right) = p(w_4|w_3) = \frac{345}{945}$$

بنابراین داریم:

$$p(w_1, w_2, \dots, w_n) = p(w_n|w_{n-1})p(w_{n-1}|w_{n-2}) \dots p(w_2|w_1)$$
$$p(w_1, w_2, \dots, w_n) = \frac{411}{1872} \times \frac{68}{2021} \times \frac{345}{945}$$

$$\rightarrow p(w_1, w_2, \dots, w_n) \approx 0.0026$$

سوال دوم

رابطهٔ داده شده، مربوط به احتمال رخداد یک جمله با n کلمه است. برای اثبات این رابطه از قضیهٔ Chain Rule استفاده می‌کنیم:

$$p(w_1^n) = p(w_1, w_2, \dots, w_n) = p(w_1)p(w_2|w_1)p(w_3|w_2, w_1) \dots p(w_n|w_1, w_2, w_3, \dots, w_{n-1})$$

$$\rightarrow\ p(w_1^n) = \prod_{k=1}^{n} p\left(w_k|w_1^{k-1}\right)$$

سوال سوم

بخش اول در گام زمانی اول، پس از Start دو کلمهٔ neural و network تولید می‌شوند و به دو مسیر زیر می‌رسیم:

1. START, neural
2. START, network

برای محاسبهٔ میزان امتیاز مسیر با استفاده از log-probabilityهای داده شده، تمام اعداد موجود در گره‌های مسیر را با هم جمع می‌کنیم. بنابراین داریم:

مسیر	امتیاز
START, neural	**-0.65**
START, network	**-0.73**

از آنجا که $k=2$ بوده و تنها دو مسیر داریم، پس هر دو مورد انتخاب شده گسترش داده می‌شوند.

بخش دوم در گام زمانی دوم، هر دو گرهٔ بدست آمده در گام پیشین را گسترش می‌دهیم، و به مسیرهای زیر می‌رسیم:

مسیر پیشین	مسیر	امتیاز
START, neural	START, neural, neural	**(-0.65) + (-0.8) = -1.45**
	START, neural, network	**(-0.65) + (-0.6) = -1.25**
START, network	START, network, neural	**(-0.73) + (-0.6) = -1.33**
	START, network, network	**(-0.73) + (-0.8) = -1.53**

از آنجا که $k=2$ است، به ازای هر مسیر اولیه، دو مورد برتر از نظر امتیاز را انتخاب کرده و در میان مسیرهای بدست آمده دو مسیر برتر را انتخاب می‌کنیم. بنابراین مسیرهای زیر در نهایت انتخاب خواهند شد:

1. START, neural, network
2. START, network, neural

بخش سوم

مسیر پیشین	مسیر	امتیاز
START, neural, network	START, neural, network, neural	**(-1.25) + (-0.8) = -2.05**
	START, neural, network, network	**(-1.25) + (-0.6) = -1.85**
START, network, neural	START, network, neural, neural	**(-1.33) + (-0.8) = -2.13**
	START, network, neural, network	**(-1.33) + (-0.6) = -1.93**

با توجه به مقادیر به دست آمده، دو مسیر برتر برای گسترش به صورت زیر خواهند بود:

1. START, neural, network, network
2. START, network, neural, network

بخش چهارم خیر؛ بهترین مسیر با در نظر گرفتن احتمالات log-probability مسیر زیر خواهد بود:

START, neural, neural, neural

که امتیاز آن برابر با 1.46- است. اما این مسیر با توجه به پرتوی ۲ انتخاب شده برای الگوریتم، و نبود این مسیر در بهترین دو مسیرها، در گام دوم حذف شده و مسیر آن ادامه نیافته است.

بخش چهارم

- در RNN به منظور تولید هر کلمه در مسیر تولید کلمات، نیاز است که به ازای هر یک از لغات موجود در لیست دیکشنری یک احتمال را به دست آوریم که همان احتمال $p(w|s)$ است که s جملهٔ تولید شده تا کنون بوده و w لغتی است که می‌خواهیم احتمال وقوع آن را در چنین جمله‌ای حساب کنیم. بنابراین در این مرحله، نیاز است که M احتمال را محاسبه کرده و مرتبهٔ محاسباتی بدست آمده $O(M)$ خواهد شد.
- با بدست آوردن M مقدار احتمالاتی، با توجه به سودوکد داده شده، در گام بعدی نیاز است که مرتب‌سازی مقادیر بدست آمده را انجام دهیم. برای مرتب‌سازی می‌توان از Quick Sort یا Merge Sort استفاده کرد که هر کدام باعث رسیدن به مجموع پیچیدگی $O(MLogM)$ می‌شود. که پس از ترکیب شدن با گام پیشین به پیچیدگی $O(MLogM)$ می‌رسیم.
- با استفاده از Beam Search نیاز است که k کپی از RNN بگیریم. به عبارتی نیاز است که مراحل بالا را به ازای هر k انتخاب در هر مرحله انجام دهیم. با ترکیب این عبارت و عبارت بالایی به پیچیدگی محاسباتی $O(kM)$ می‌رسیم.
- پس از اجرای کدهای بالا، به k^2 دنباله می‌رسیم که باز نیاز است که آن‌ها را مرتب کنیم و از میانشان k مورد را انتخاب کنیم. برای سورت کردن به پیچیدگی زمانی $O(k^2Log(k^2)) = O(2k^2Log(k)) = O(k^2Log(k))$ می‌رسیم. با ترکیب کردن این پیچیدگی و پیچیدگی پیشین و این نکته که $M \geq k$ است به پیچیدگی $O(kMLogM)$ می‌رسیم.
- پس از اجرای کدای بالا در نهایت، یک تکرار به ازای T گام زمانی خواهیم داشت تا دنباله‌ای به طول T در انتها بدست آید. بنابراین پیچیدگی زمانی نهایی به صورت $O(TkMLogM)$ است.

سوال چهارم

بخش اول در ابتدا هر یک از گیت‌ها را توضیح داده و سپس عواقب و نیز نتیجهٔ بدست آمده از معماری تازه برای *LSTM* را شرح می‌دهم.

- *Forget Gate*: در این گیت، میزان تاثیر اهمیت حافظهٔ بلند مدت مدل را بدست می‌آوریم. به عنوان مثال در صورت ورود به یک جملهٔ جدید می‌توانیم وزن‌های مربوط به این گیت را به نحوی تعیین کنیم تا تاثیر حافظهٔ بلند مدت را صفر کند.
- *Input Gate*: در این گیت، مشخص می‌کنیم که تا چه حد توکن خوانده شده در گام زمانی حال حاضر در خروجی تاثیرگذار است. به عبارتی با استفاده از این گیت مشخص می‌کنیم که کلمهٔ تازه خوانده شده تا چه میزان بر روی نتیجهٔ نهایی تاثیرگذار است.
- *Output Gate*: با استفاده از این گیت می‌توانیم جریان خروجی هر یک بلوک‌های *LSTM* را کنترل کنیم و مشخص کنیم که حافظهٔ کوتاه مدت به چه نحوی بروزرسانی شود.

حال در صورتی که تنها *Forget Gate* را نگه داریم، و دو گیت ذکر شدهٔ دیگر را حذف کنیم، باعث بوجود آمدن یک معماری تازه خواهیم شد؛ به صورتی که تنها تاثیر حافظهٔ بلند مدت در تعیین خروجی تاثیرگذار بوده و باعث وقوع عواقب زیر می‌شود:

- بدون *Input Gate*، شبکه توانایی کنترل جریان اطلاعات جدید بدست آمده از توکن‌های ورودی در هر گام زمانی را از دست می‌دهد. این بدان معنی است که تمام ورودی های جدید بدون توجه به ارتباط آن به سلول اضافه می شود.
- بدون *Output Gate*، شبکه نمی‌تواند جریان اطلاعات مربوط به حافظهٔ کوتاه مدت بدست آمده از یک بلوک *LSTM* را تنظیم کند. کل حافظهٔ کوتاه مدت بدست آمده در هر بلوک مستقیماً در اختیار لایهٔ بعدی قرار می‌گیرد و آن را با اطلاعات نامرتبط اشباع می‌کند.

همچنین نتیجه‌ای که ممکن است به وقوع بپیوندد به شرح زیر است:

> خروجی احتمالاً بسیار ناپایدار و به طور بالقوه بی معنی خواهد شد. شبکه به طور مداوم اطلاعات جدیدی را بدون کنترل اضافه می‌کند، که منجر به ایجاد مداوم داده‌های نامربوط در هر بلوک *LSTM* می‌شود.

بخش دوم در چنین حالتی، توانایی شبکه در به یاد داشتن اطلاعات قدیمی از بین می‌رود؛ چرا که در هر گام زمانی، میزان تاثیرگذاری حافظهٔ بلند مدت را صفر کرده و در نتیجه خروجی و محاسبات در هر گام زمانی تنها متکی بر ورودی در این گام زمانی و نیز حافظهٔ کوتاه مدت بدست آمده از گام زمانی پیشین است. این موضوع به طور بالقوه می‌تواند منجر به شکست شبکه در حفظ وابستگی‌های طولانی مدت و الگوهای حافظهٔ ضروری برای کارهایی مانند *Language Translation* یا *Language Model* شود. بنابراین، توانایی شبکه برای یادگیری و پیش‌بینی ممکن است به شدت آسیب ببیند، به خصوص در کارهایی که حفظ اطلاعات طولانی مدت حیاتی است. به عنوان مثال ممکن است که در یک جمله برای تعیین فعل، نیاز به فاعلی داشته باشیم که در ابتدای جمله آورده شده است. در چنین حالتی و با صفر شدن میزان تاثیر حافظهٔ بلند مدت، ممکن است که شبکهٔ عصبی در تعیین فعل و ضمیر آن دچار اشتباه شود.

بخش سوم

- مزایا: افزایش تعداد لایه های *LSTM* در یک شبکه، که اغلب به عنوان *LSTM* های انباشته نامیده می‌شود، می‌تواند منجر به بهبود در یادگیری الگوهای پیچیده و نمایش سلسله مراتبی عمیق‌تر داده‌های موجود در یک توالی شود. هر لایه اضافی به طور بالقوه می‌تواند سطحی از انتزاع یا پیچیدگی را به نمایش بدست آمده از ورودی اضافه کند و ساختارهای پیچیده‌تری را در داده‌ها بدست آورد.
- معایب: با این حال، افزودن لایه‌های *LSTM* بیشتر، پیچیدگی شبکه را هم از نظر نیازهای محاسباتی و هم از نظر سختی آموزش افزایش می‌دهد. لایه‌های بیشتر می‌تواند منجر به زمان آموزش طولانی‌تر شود و به منابع محاسباتی بیشتری نیاز داشته باشد. علاوه بر این، شبکه‌های عمیق‌تر اغلب مستعد برازش بیش از حد (*Overfitting*) هستند، به‌ویژه در مجموعه داده‌های کوچک‌تر، و ممکن است برای تعمیم خوب به تکنیک‌های منظم‌سازی پیچیده‌تری نیاز داشته باشند و یا مجموعهٔ دادهٔ بزرگتری را برای آموزش ایجاد کرد.

در پایان لازم به ذکر است که علاوه بر موارد گفته شده، بازدهی کاهشی در افزودن لایه‌های بیشتر وجود دارد. فراتر از تعداد معینی از لایه‌ها، مزیت بدست آمده ممکن است افزایش پیچیدگی و هزینهٔ محاسباتی را توجیه نکند.