به نام خدا

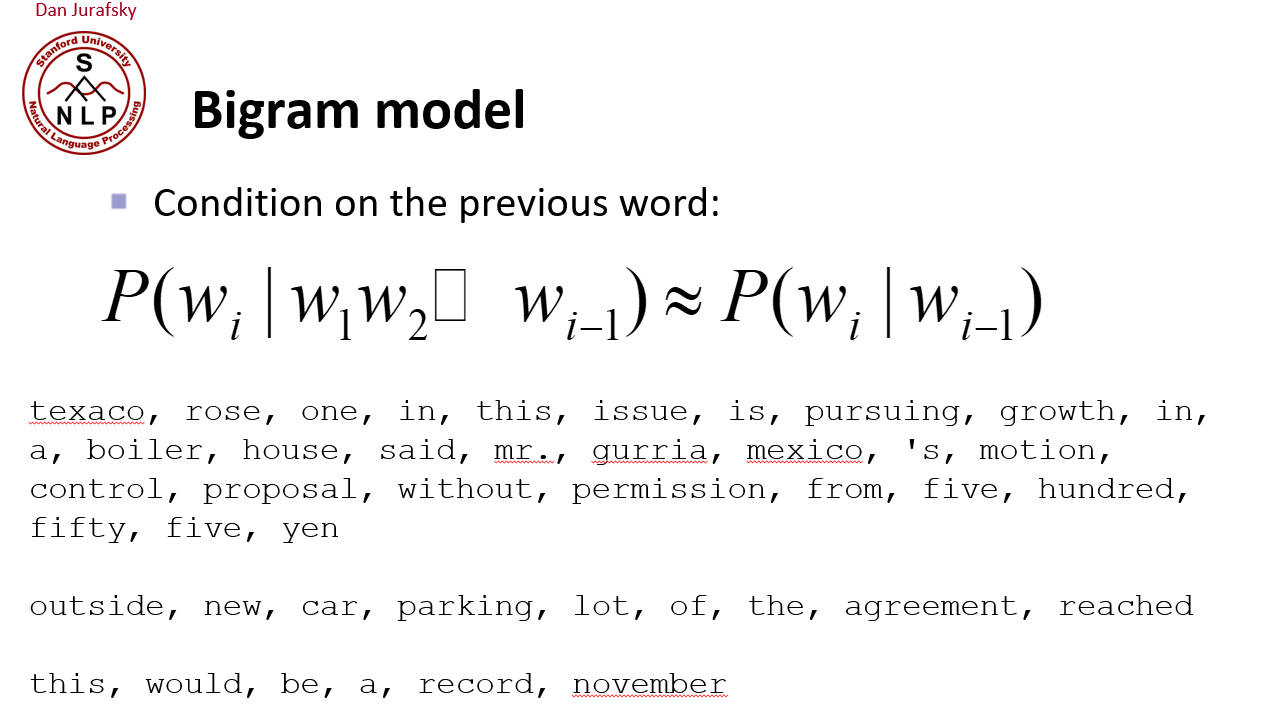
# تمرین سری چهارم درس مبانی پردازش زبان و گفتار دکتر مرضیه داودآبادی فراهانی

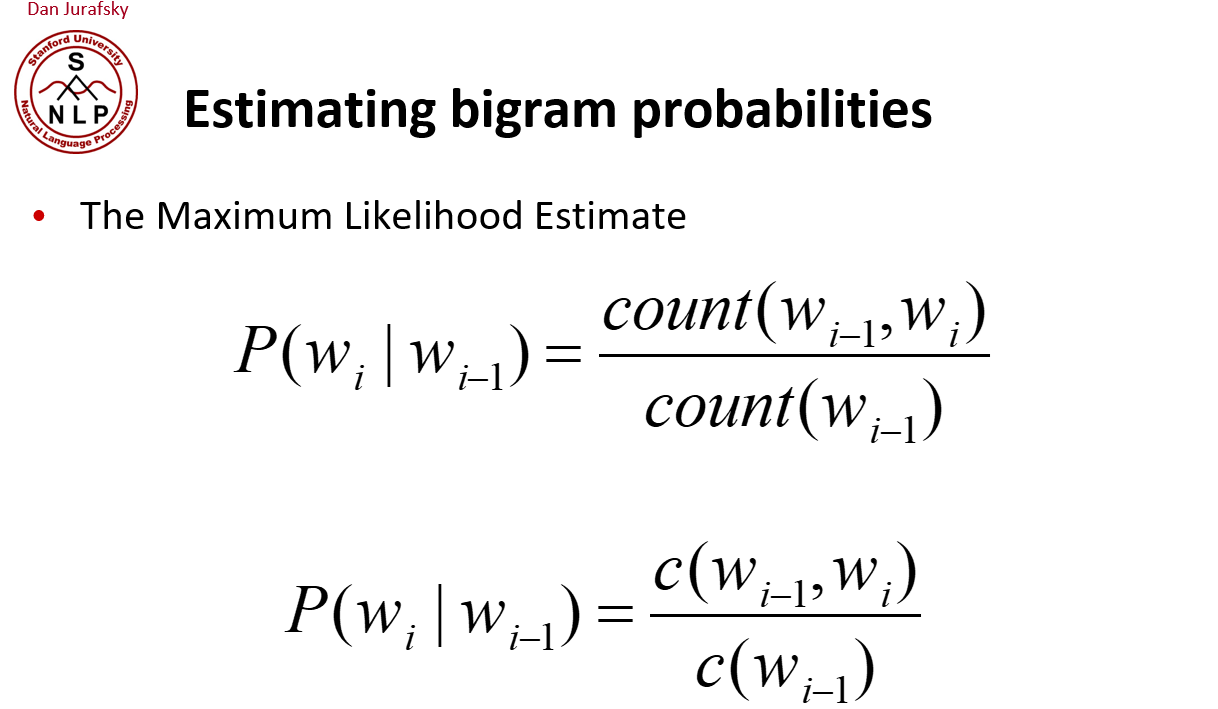
# فرزان رحمانی 99521271

# سوالات تئوری

## سوال اول

برای به دست آوردن احتمال هر جمله باید از رابطه سوال 2 استفاده کنیم. اما در اینجا جون فقط bigram ها را داریم از تقریب رابطه استفاده میکنیم. یعنی از فرض میکنیم(**Markov Assumption**) احتمال هر کلمه فقط به کلمه قبلی وابسته است و از دیگر کلمات مستقل است. پس رابطه سوال 2 تبدیل به میشود.





unigram ها را با ها و bigram ها را با نشان می دهیم.

حال با توجه به جداول صورت سوال، جدول احتمال Bigram ها را به‌کمک فرمول زیر ایجاد می‌کنیم:

(در جدول سوال فرض میکنیم که سطر نشان دهند کلمه اول و ستون نشان دهنده کلمه دوم است. پس هر سطر جدول bigram ها تقسیم بر مقدار متناظر موجود در جدول unigram ها می شود.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| کتاب | داستان | امروز | دیروز | خواندیم | ما |  |
| ۰.۲۰۶ | ۰.۱۲۷ | ۰.۲۴۱ | ۰.۲۲ | ۰.۰۸۳ | ۰ | ما |
| ۰.۰۷۴ | ۰.۰۸۸ | ۰.۰۰۷ | ۰.۰۰۴ | ۰ | ۰.۰۰۱ | خواندیم |
| ۰.۱۲۵ | ۰.۰۳۳ | ۰.۰۲۳ | ۰ | ۰.۰۱۵ | ۰.۱۶۸ | دیروز |
| ۰.۱۱۸ | ۰.۰۴۴ | ۰ | ۰ | ۰.۰۰۶ | ۰.۱۶۸ | امروز |
| ۰.۰۰۱ | ۰ | ۰.۰۴ | ۰.۰۸۸ | ۰.۳۶۵ | ۰.۰۰۴ | داستان |
| ۰ | ۰.۳۲۳ | ۰ | ۰.۰۰۲ | ۰.۲۵۷ | ۰.۰۲۴ | کتاب |

حال از رابطه استفاده میکنیم.

احتمال جمله تست 1:

P(ما امروز کتاب خواندیم) = P(خواندیم | کتاب).P(کتاب | امروز).P(امروز | ما) =

احتمال جمله تست 2:

P(ما دیروز داستان خواندیم) = P(خواندیم | داستان).P(داستان | دیروز).P(دیروز | ما) =

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

## سوال دوم

رابطه سوال قانون زنجیره ای برای محاسبه احتمال یک دنباله از کلمات در یک مدل زبان است. احتمال مشترک کل دنباله را به ضرب احتمالات شرطی تقسیم می کند، که احتمال هر کلمه به کلمات قبلی در دنباله مشروط می شود.

برای اثبات این معادله می توانیم از تعریف قانون بیز نظریه احتمال استفاده کنیم.  
  
ابتدا از قانون بیز شروع میکنیم.

پس میتوانیم بنویسیم:

لذا برای هر i داریم:

مثلا داریم:

حال با جایگذاری در و بسط دادن i های مختلف رابطه مطلوب اثبات می شود.

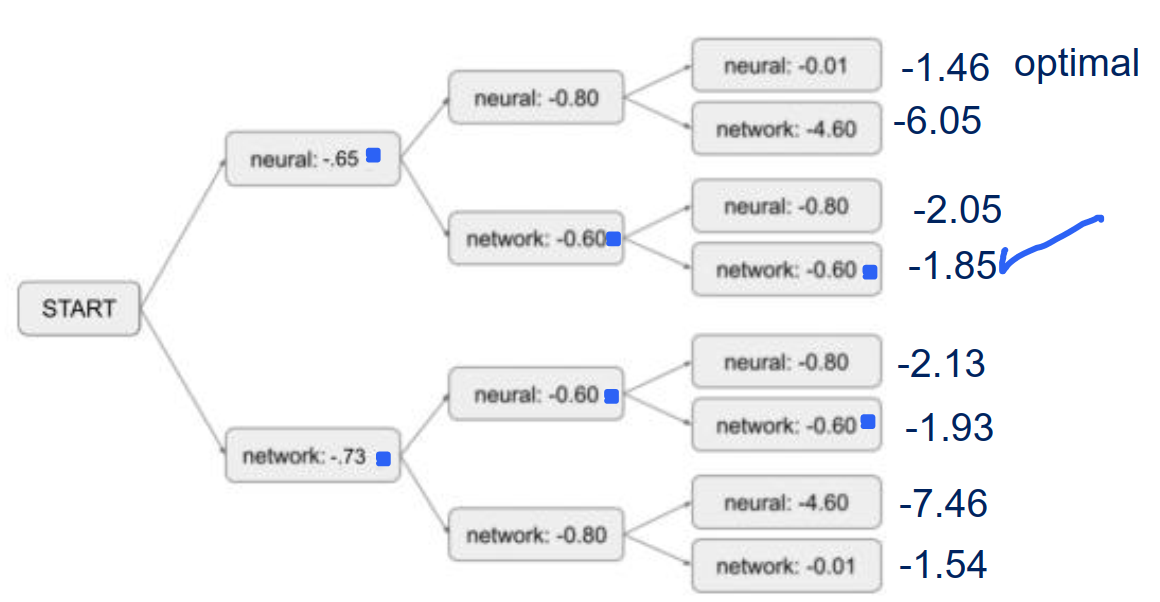
بنابراین، ما ثابت کردیم که احتمال مشترک یک دنباله از کلمات را می توان به عنوان حاصل ضرب احتمالات شرطی بیان کرد، که در آن احتمال هر کلمه به کلمات قبلی در دنباله مشروط می شود.

این معادله اساس تکنیک‌های مدل‌سازی زبان، مانند مدل‌های n-gram و مدل‌های مبتنی بر شبکه‌های عصبی را تشکیل می‌دهد که هدف آنها برآورد این احتمالات شرطی از مجموعه‌های متنی بزرگ است.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

## سوال سوم



1. در مرحله زمانی 1 دنباله های و ذخیره می شوند.

t = 1  
for each hypothesis that we are tracking find top 2 tokens (find the probabilities):

Sort the resulting sequences by their total log-probability:

Store the top-2 sequences:

1. در مرحله زمانی 2 دنباله های و ذخیره می شوند.

t = 2  
for each hypothesis that we are tracking find top 2 tokens (find the probabilities):

Sort the resulting sequences by their total log-probability:

Store the top-2 sequences:

1. در مرحله زمانی 3 دنباله های و ذخیره می شوند.

t = 3  
for each hypothesis that we are tracking find top 2 tokens (find the probabilities):

Sort the resulting sequences by their total log-probability:

Store the top-2 sequences:

*بنابراین جواب نهایی و خروجی الگوریتم دنباله* خواهد بود.

1. خیر، beam search دنباله ی کلی با بیشترین احتمال در این مثال را بر نمی گرداند. به بیان دیگر الگوریتم beam search بهینه ترین جواب را بر نمیگرداند و الگوریتم optimal نیست. چرا که در این مثال خروجی این الگوریتم دنباله *با log probability = -1.85 است ولی جواب بهینه   
   (overall most-likely sequence) دنباله* با  
    *log probability = -0.65 – 0.8 -0.01 = -1.46 است.*این دنباله‌ برگردانده نمی‌شوند زیرا در مرحله 2 از بررسی حذف می‌شود، از آنجایی که "network network" در دنباله  
    k = 2 دنباله با بیشترین احتمال طول-2 نیست.

*توضیحات بیشتر:*الگوریتم beam search لزوماً توالی محتمل‌ترین توالی را در یک مدل زبان رمزگشایی برمی‌گرداند. در عوض، هدف آن یافتن یک تقریب خوب از محتمل‌ترین دنباله به شیوه‌ای کارآمد است.

وظیفه یافتن محتمل‌ترین توالی به دلیل رشد تصاعدی تعداد دنباله‌های ممکن با افزایش طول، از نظر محاسباتی برای دنباله‌های طولانی غیرقابل حل است. ارزیابی جامع(exhaustive search) همه دنباله‌های ممکن برای یافتن محتمل‌ترین توالی واقعی امکان‌پذیر نیست، به‌ویژه برای اندازه‌های بزرگ واژگان و دنباله‌های طولانی.

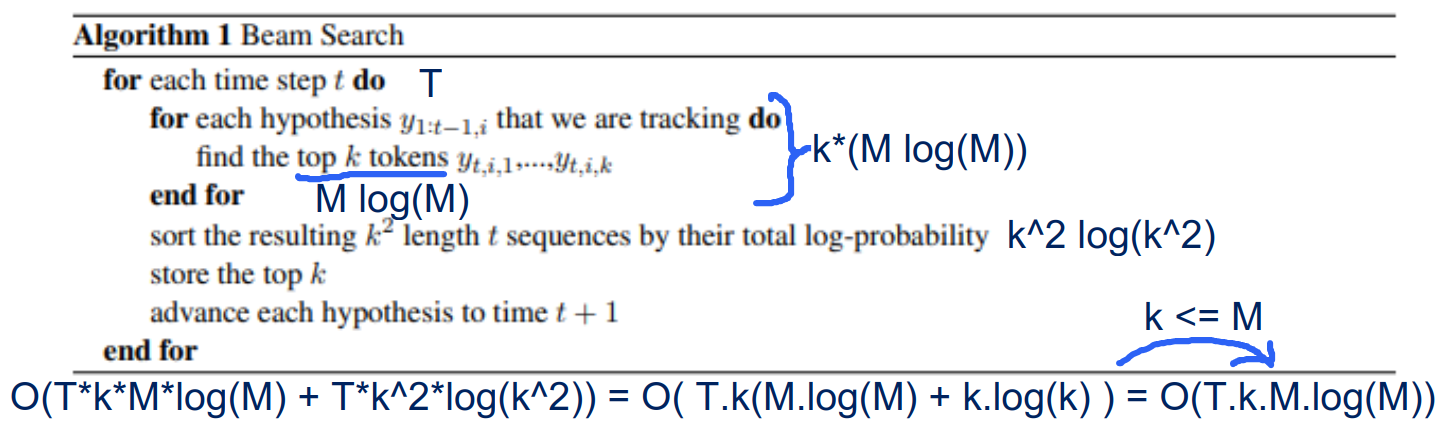
الگوریتم beam search یک استراتژی جستجوی اکتشافی است که مجموعه محدودی از فرضیه‌های جزئی (توالی) امیدوارکننده را در هر مرحله بررسی می‌کند و احتمال کمتری برای کنترل فضای جستجو را حذف می‌کند. به طور خاص، در هر مرحله زمانی، الگوریتم k فرضیه‌های جزئی محتمل‌تر (که k عرض پرتو است) را دنبال می‌کند و آنها را با یک کلمه گسترش می‌دهد، فرضیه‌های جدید را امتیاز می‌دهد و فقط top-k را برای مرحله بعدی نگه می‌دارد.

الگوریتم جستجوی پرتو(beam search) با جستجوی یک زیرمجموعه محدود از کل فضای جستجو، دقت را با کارایی کاهش می‌دهد. با نگه داشتن فقط k فرضیه های بالا در هر مرحله، به طور موثر بخش بزرگی از فضای جستجو را حذف می کند و فرآیند رمزگشایی را از نظر محاسباتی قابل پردازش می کند.

با این حال، این هرس همچنین به این معنی است که احتمالاً دنباله واقعی ممکن است دور انداخته شود اگر در نقطه‌ای در طول جستجو در خارج از پرتو قرار گیرد. الگوریتم فقط می تواند یافتن محتمل ترین دنباله را در فضای جستجوی محدود تعریف شده توسط عرض پرتو تضمین کند.

در عمل، جستجوی پرتو با پهنای پرتویی که به دقت انتخاب شده است، اغلب می‌تواند تقریب‌های خوبی از محتمل‌ترین توالی پیدا کند، به‌ویژه برای اندازه‌های نسبتاً کوچک واژگان و دنباله‌های کوتاه تا متوسط. با این حال، برای واژگان بسیار بزرگ و دنباله‌های طولانی، کیفیت تقریب ممکن است کاهش یابد، و الگوریتم ممکن است به دلیل هرس تهاجمی، توالی محتمل واقعی را از دست بدهد.

به طور خلاصه، الگوریتم beam search یک روش اکتشافی کارآمد برای رمزگشایی در مدل‌های زبانی است، اما یافتن محتمل‌ترین توالی کلی را تضمین نمی‌کند. در عوض، هدف آن یافتن تقریب خوب با کاوش در یک زیرمجموعه محدود از فضای جستجوی تعریف شده توسط عرض پرتو است.

1.   
     
   پیچیدگی زمان اجرا تولید یک دنباله با طول Tبا اندازه پرتو k با RNN برابر با O( T.k.M.log(M) ) است. در بالا روی سودو کد توضیح داده شده همچنین دلایل بیشتر در ادامه آمده است.

• Step RNN forward one step for one hypothesis = *O*(*M*) (since we compute one logit for each vocab item, and none of the other RNN operations rely on *M*, *T* , or *K*).

• Do the above, and select the top *k* tokens for one hypothesis. We do this by sorting the logits: *O*(*M* log *M*). (Note: there are more efficient ways to select the top *K*, for instance using a min heap. We just use this way since the code implementation is simple.) Combined with the previous step, this is *O*(*M* log *M* + *M*) = *O*(*M* log *M*)*.*

• Do the above for all *k* current hypotheses *O*(*KM* log *M*).

• Do all above + choose the top *K* of the hypotheses currently stored: we do this by sorting the array of items: *O*( log() = *O*( log(*K*)) (since log = 2 log(*K*)). (Note: there are also more efficient ways to do this). Combining this with the previous steps, we get *O*(*KM* log *M* + log(*K*)). When one term is strictly larger than another we can take the max of the two. Since *M ≥ K*, we could also write this as *O*(*KM* log *M*).

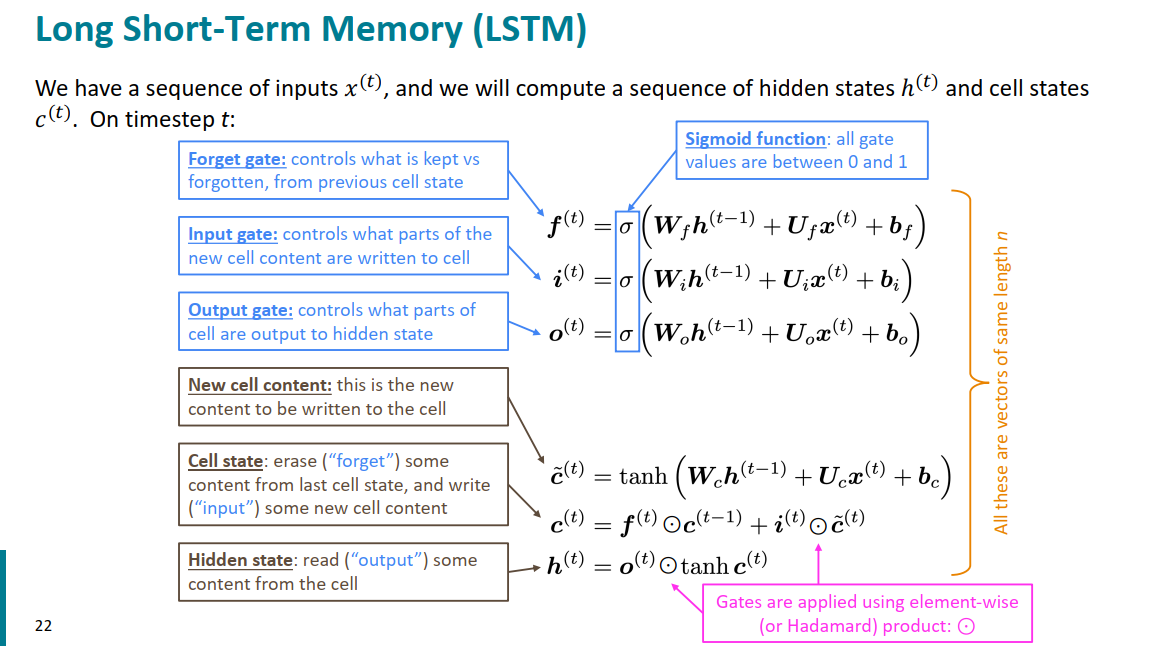
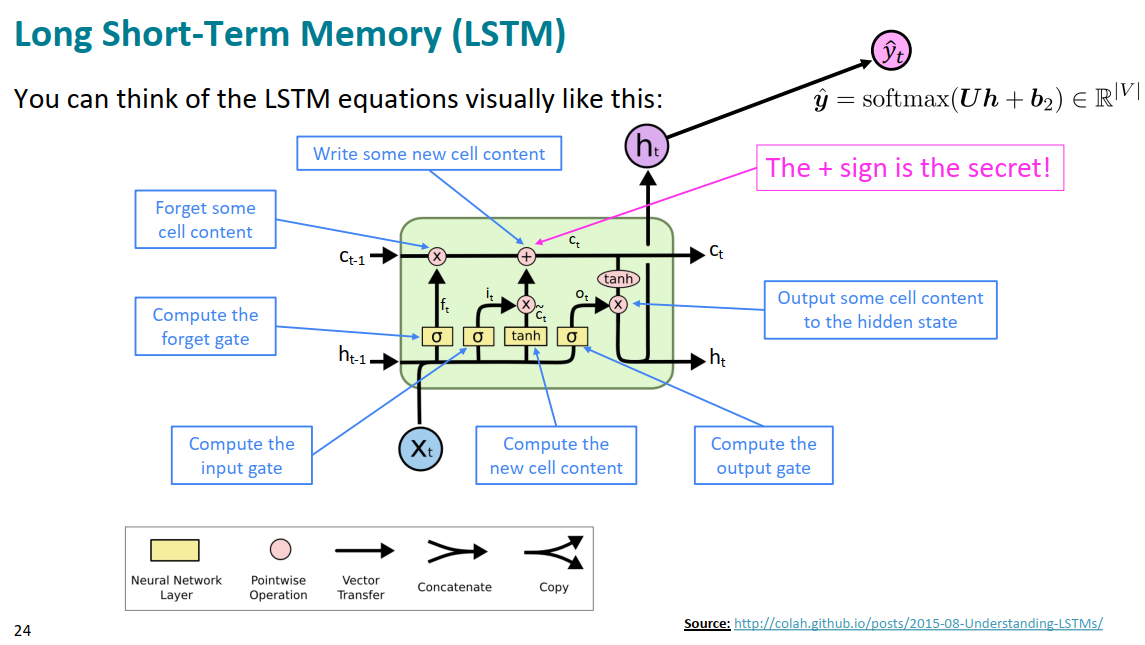
• Repeat this for *T* timesteps: *O*(*TKM* log *M*).

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>   
<https://inst.eecs.berkeley.edu/~cs182/sp23/assets/assignments/hw6_sol.pdf>

## سوال چهارم

ابتدا مرور میکنیم که LSTM چه بود.

1. اگر فقط گیت فراموشی را در LSTM داشته باشیم و گیت های ورودی و خروجی را حذف کنیم، اساساً LSTM را به یک شبکه عصبی بازگشتی ساده (RNN) با سلول های حافظه گیت دار تبدیل می کنیم. دروازه فراموشی کنترل می کند که چه مقدار از حالت سلول قبلی باید حفظ یا فراموش شود. با این حال، بدون گیت ورودی، هیچ مکانیزمی برای افزودن اطلاعات جدید به حالت سلول وجود نخواهد داشت و بدون گیت خروجی، کل حالت سلول در هر مرحله زمانی خروجی خواهد بود.  
   این کار به طور قابل توجهی توانایی LSTM را برای کنترل انتخابی اطلاعاتی که باید به وضعیت سلول اضافه یا حذف شود، و همچنین آنچه که باید به عنوان خروجی در معرض دید قرار گیرد، محدود می کند. خروجی احتمالاً noisy و کم معنا خواهد بود، زیرا به سادگی حالت سلولی انباشته شده در طول زمان بدون هیچ گونه فیلتر یا کنترل اضافی خواهد بود.

* The input gate in an LSTM determines how much of the new information should be stored in the cell state, while the output gate controls how much information from the cell state should be exposed to the rest of the network.
* If we remove both the input and output gates, the LSTM essentially becomes a modified version of the vanilla RNN (Recurrent Neural Network) where the cell state is solely controlled by the forget gate.
* In this scenario, the LSTM would have limited capacity to retain useful information over long sequences because it lacks mechanisms to control what information is stored and how it's used.
* The output would likely suffer in tasks where long-term dependencies are crucial, as the LSTM wouldn't be able to effectively retain relevant information over time.

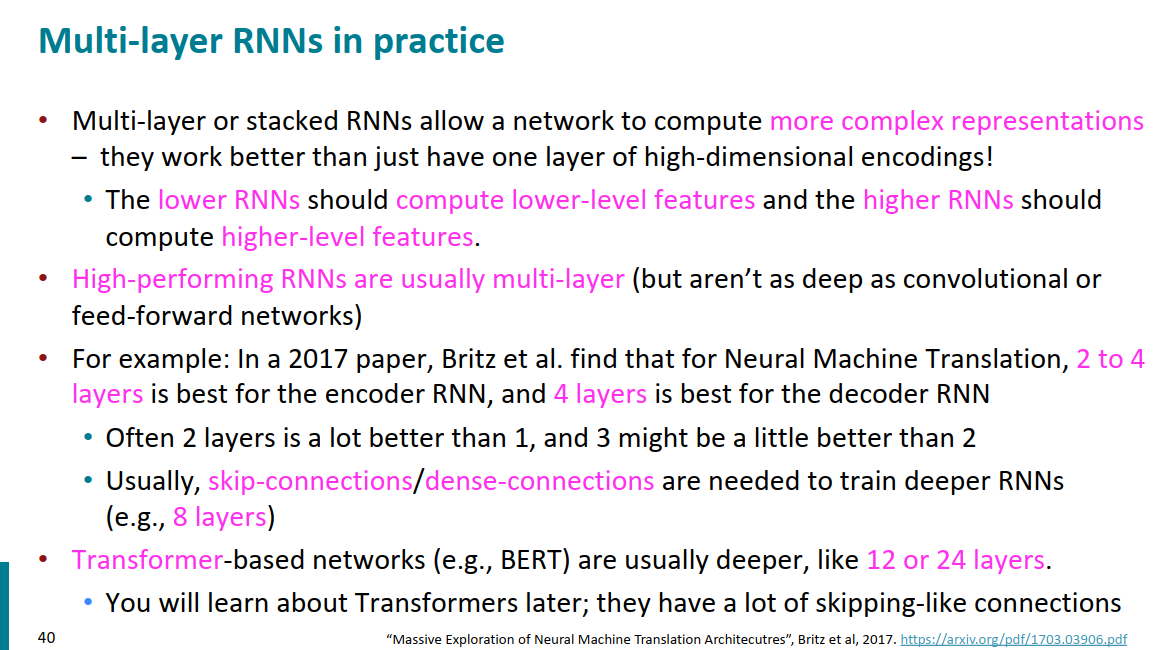
1. اگر مقدار گیت فراموشی را در یک LSTM بر روی صفر قرار دهیم، به این معنی است که LSTM اساساً کل وضعیت سلول را از مرحله زمانی قبلی "فراموش می کند" یا دور می زند. این امر شبکه را از حفظ و استفاده از هرگونه وابستگی بلندمدت یا اطلاعات زمینه ای از گذشته جلوگیری می کند.

بدون توانایی انتقال اطلاعات مربوطه از مراحل زمانی قبلی، LSTM برای یادگیری و مدل‌سازی موثر داده‌های متوالی مشکل دارد. نمی تواند وابستگی های دوربرد را جذب کند و از آن استفاده کند، که یکی از نقاط قوت کلیدی LSTM در مقایسه با RNN های سنتی است.

علاوه بر این، با تنظیم گیت فراموشی روی صفر، LSTM توانایی حفظ یا فراموشی انتخابی بخش‌هایی از حالت سلول را از دست می‌دهد. این کنترل جزئی بر روی اینکه چه اطلاعاتی باید حفظ یا دور ریخته شود برای LSTM برای یادگیری مؤثر و پیش‌بینی دقیق بسیار مهم است، به‌ویژه در کارهایی که شامل توالی‌های پیچیده با وابستگی‌های طولانی‌مدت است.

به طور کلی، تنظیم مقدار گیت فراموشی روی صفر به میزان قابل توجهی ظرفیت LSTM برای یادگیری و پیش بینی را مختل می کند، زیرا به طور موثر آن را به یک مدل بدون حافظه کاهش می دهد که نمی تواند از اطلاعات گذشته استفاده کند.

* The forget gate controls how much of the previous cell state to forget. Setting it to zero essentially means that the LSTM would completely forget the previous cell state.
* In this case, the LSTM would not retain any information from the previous time step, leading to a severe loss of context and memory. It would essentially behave like starting anew at each time step, disregarding any historical information.
* This would severely impair the LSTM's ability to learn and predict, especially in tasks where understanding context over time is crucial, such as language modeling or sequential data processing.

1. ابتدا مطالب بیان شده در اسلاید های درس را بیان میکنیم سپس به توضیح می پردازیم:  
     
     
   افزایش تعداد لایه های LSTM در یک شبکه به طور بالقوه می تواند کارایی و عملکرد شبکه را بهبود بخشد، اما پیچیدگی و نیازهای محاسباتی را نیز افزایش می دهد.

داشتن چندین لایه LSTM به شبکه اجازه می دهد تا الگوها و نمایش های پیچیده تری را در سطوح مختلف انتزاع بیاموزد و ثبت کند. هر لایه می‌تواند بر بازنمایی‌های آموخته‌شده توسط لایه قبلی ساخته شود، و به شبکه اجازه می‌دهد تا ویژگی‌های سطح بالاتر و انتزاعی‌تر داده‌های ورودی را مدل‌سازی کند.

این یادگیری نمایش سلسله مراتبی می‌تواند به ویژه برای کارهایی که شامل توالی‌های طولانی یا ساختارهای داده پیچیده است مفید باشد، زیرا لایه‌های بالاتر می‌توانند وابستگی‌های دوربرد و الگوهای جهانی را به طور مؤثرتری نسبت به یک لایه واحد ثبت کنند.

با این حال، افزایش تعداد لایه‌های LSTM تعداد پارامترهای شبکه را نیز افزایش می‌دهد، که می‌تواند فرآیند آموزش را از نظر محاسباتی فشرده‌تر کند و در صورت عدم تنظیم صحیح، خطر بیش از حد برازش را افزایش دهد.

علاوه بر این، آموزش شبکه‌های عمیق‌تر به دلیل مسائلی مانند ناپدید شدن یا انفجار گرادیان‌ها می‌تواند چالش‌برانگیزتر باشد، که می‌تواند با تکنیک‌هایی مانند برش گرادیان، مقداردهی اولیه دقیق و معماری‌های تخصصی مانند اتصالات باقی‌مانده کاهش یابد.

به طور خلاصه، در حالی که افزایش تعداد لایه‌های LSTM به طور بالقوه می‌تواند توانایی شبکه را برای گرفتن الگوهای پیچیده و وابستگی‌های دوربرد بهبود بخشد، همچنین پیچیدگی محاسباتی را افزایش می‌دهد و چالش‌های بیشتری را در آموزش و بهینه‌سازی معرفی می‌کند. تعداد بهینه لایه ها اغلب به وظیفه خاص، پیچیدگی داده ها و منابع محاسباتی موجود بستگی دارد.

* Adding more LSTM layers to a network can enhance its capacity to capture complex patterns and dependencies in the data.
* Each additional layer allows the network to learn increasingly abstract representations of the input data.
* Deeper architectures can capture hierarchical features, enabling the network to model more intricate relationships within the data.
* However, increasing the number of LSTM layers also introduces additional parameters, which can lead to increased computational complexity and training time.
* Moreover, deeper networks are more prone to overfitting, so proper regularization techniques such as dropout or batch normalization may be necessary to prevent this.
* Overall, increasing the number of LSTM layers can improve the efficiency and performance of the network by enabling it to learn more sophisticated representations of the input data, but it also introduces complexities that need to be carefully managed.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

# سوالات عملی

## سوال پنجم

جواب این سوال در نوتبوک Q5.ipynb آمده است.  
  
مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

پایان