

Semana 16 - Modelos de Atenção

João Florindo

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas - Brasil
florindo@unicamp.br

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Modelo Básico

- Modelos sequência-para-sequência são úteis em aplicações como tradução e reconhecimento de fala
- EX.:
 - “Jane visite l’Afrique em septembre.”
 - “Jane is visiting Africa in September.”
- Representamos por $x^{<i>}$ as palavras na sentença de entrada e por $y^{<i>}$ as palavras na saída
- Como podemos treinar uma rede que recebe a sentença x e retorna y ?¹

¹Sutskever et al., 2014. Sequence to sequence learning with neural networks
Cho et al., 2014. Learning phase representations using RNN encoder-decoder for statistical machine translation

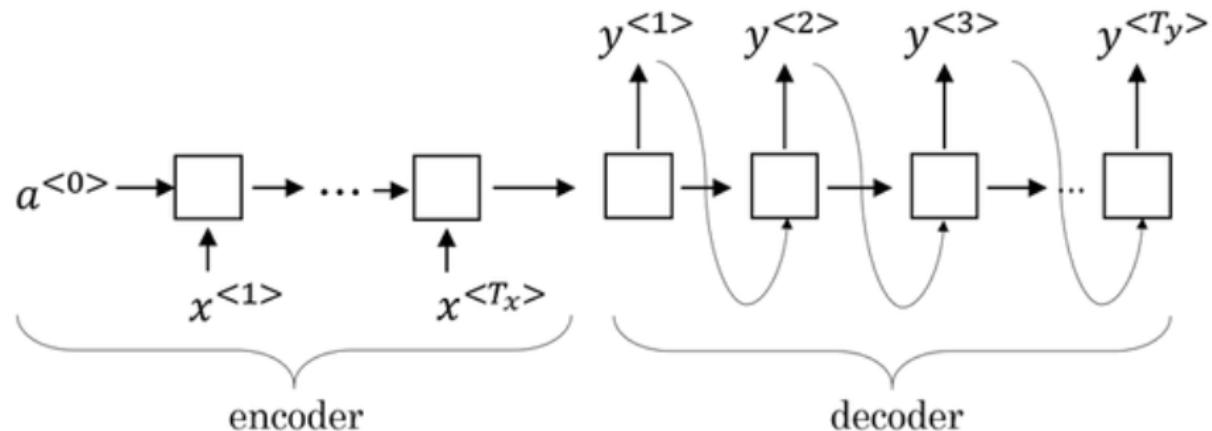
Modelo Básico

$$x^{<1>} \quad x^{<2>} \quad x^{<3>} \quad x^{<4>} \quad x^{<5>}$$

Jane visite l'Afrique en septembre

→ Jane is visiting Africa in September.

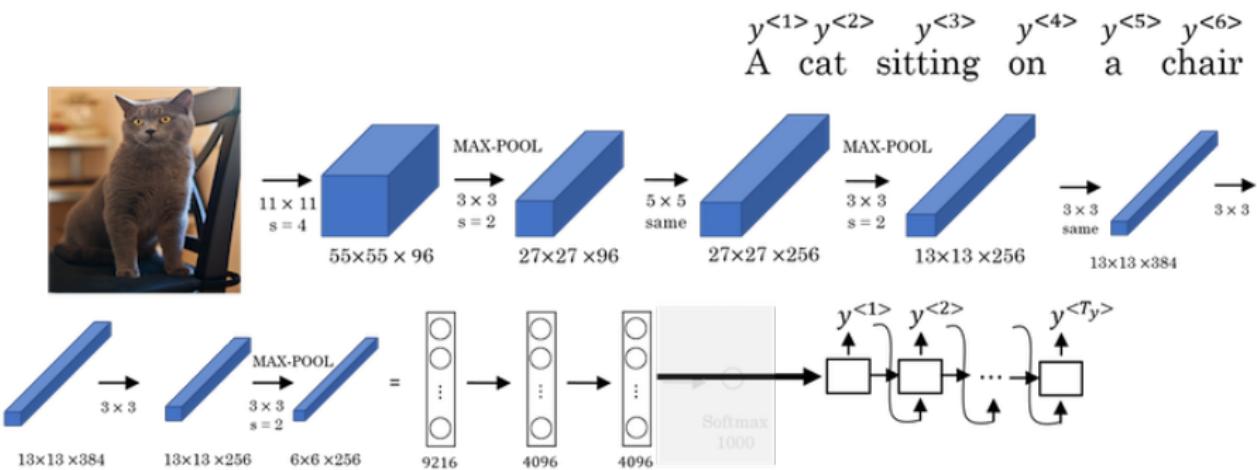
$$y^{<1>} \quad y^{<2>} \quad y^{<3>} \quad y^{<4>} \quad y^{<5>} \quad y^{<6>}$$



Modelo Básico

- Autores: Ilya Sutskever, Vinyals Oriol e V. Le. Quoc Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk e Yoshua Bengio
- *Encoder* (RNN, GRU ou LSTM) recebe a sentença em francês e tem como saída um vetor que representa a sentença inteira
- O *decoder* recebe a saída do *encoder* e é treinado para devolver a sentença traduzida, até gerar o *token* ¡EOS¿
- E, como na síntese de texto, a entrada em cada tempo é o *token* de saída do passo anterior
- Dado um conjunto razoável de sentenças em francês e inglês, este modelo funciona bem!

Geração de Legenda²



²Mao et al., 2014. Deep captioning with multimodal recurrent neural networks

Geração de Legenda

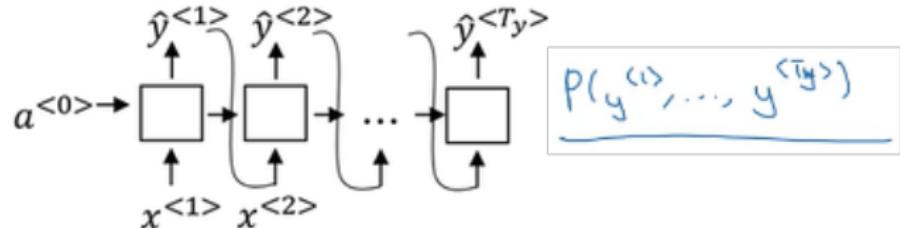
- Podemos passar uma imagem por uma ConvNet pré-treinada e aprender um *encoding* tomando a saída da penúltima camada
- A rede pré-treinada pode funcionar como o *encoder* e o vetor de 4096 componentes na AlexNet representa a imagem
- O *encoding* alimenta uma RNN, responsável por gerar a legenda
- Funciona bem, especialmente se a legenda não for muito longa
- Autores: Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, Alan Yuille

Outline

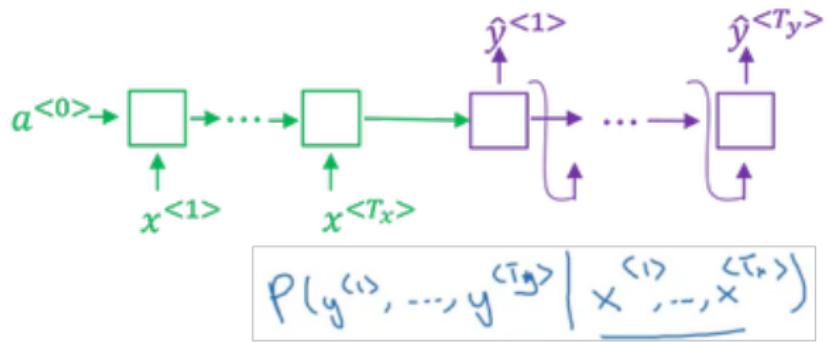
- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Sentença Mais Provável

Language model:



Machine translation:



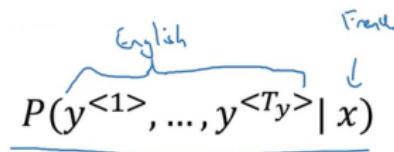
Sentença Mais Provável

- Ao contrário da síntese de textos, agora não queremos mais uma sequência aleatória de palavras, mas sim a tradução ou legenda mais provável
- Temos agora um modelo de linguagem **condicionado**
- Note que o *decoder* é idêntico ao modelo de linguagem
- Porém, em vez de começar com um vetor $a^{<0}$ de zeros, ele parte de uma representação da sentença de entrada gerada pelo *encoder*
- O modelo de linguagem dava a probabilidade de uma sentença
- Já a rede de tradução modela a probabilidade de uma sentença em inglês dada a sentença em francês (“condicionado”)

Sentença Mais Provável

- Gerar uma sentença aleatória como na síntese de texto não é mais suficiente
- Objetivo é maximizar a probabilidade condicionada pela sentença de entrada (*beam search*)

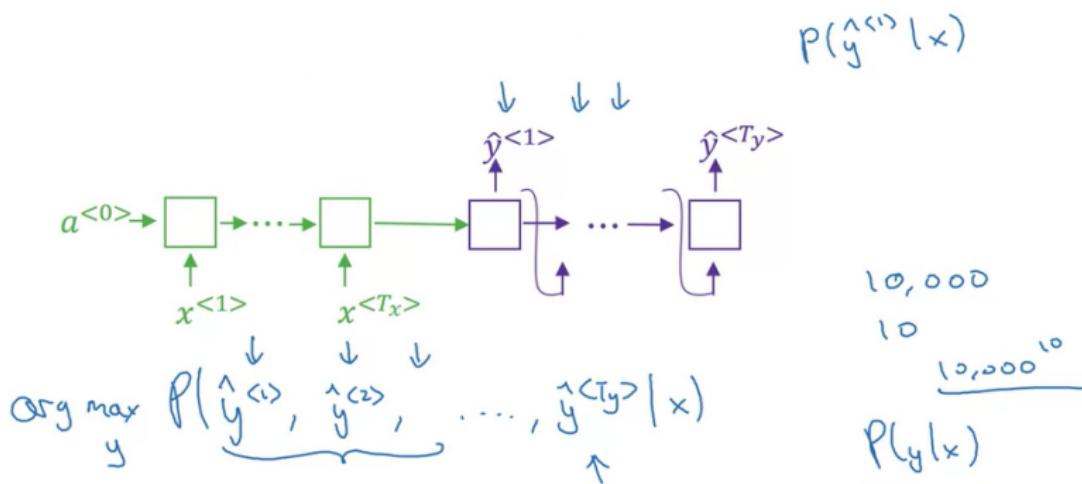
Jane visite l'Afrique en septembre.



- Jane is visiting Africa in September.
- Jane is going to be visiting Africa in September.
- In September, Jane will visit Africa.
- Her African friend welcomed Jane in September.

$$\arg \max_{y^{<1>} \dots, y^{<T_y>}} P(y^{<1>} \dots, y^{<T_y>} | x)$$

Greedy Search



- Jane is visiting Africa in September.
 - Jane is going to be visiting Africa in September.
- $P(\text{Jane is going } | x) > P(\text{Jane is visit } | x)$

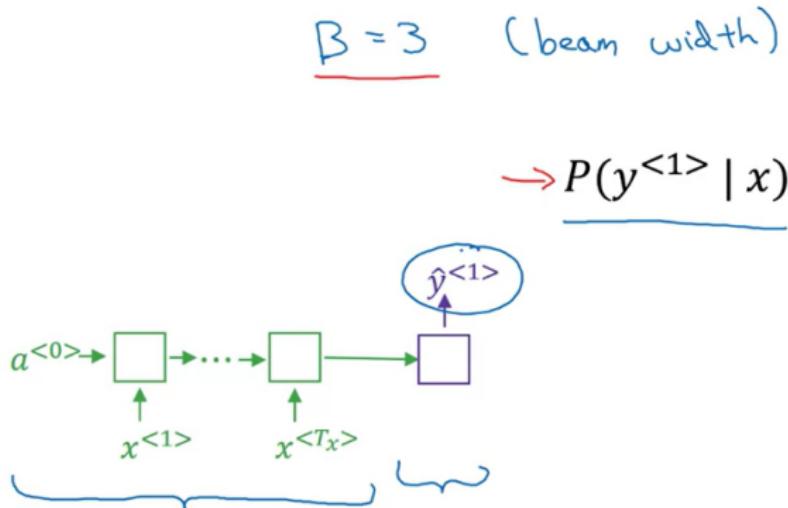
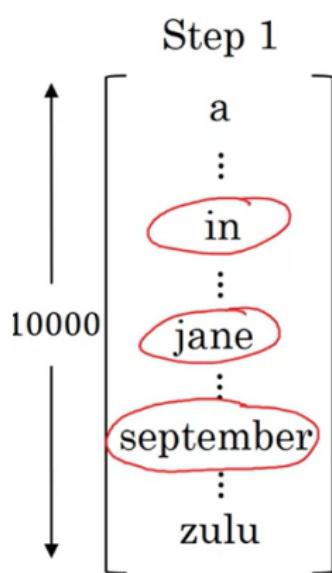
Greedy Search

- O *greedy search* amostra em cada passo de tempo a palavra mais provável naquele momento
- Mas queremos amostrar a **sentença inteira** com a maior probabilidade conjunta
- Como “going” é mais comum em inglês do que “visiting”, ela acaba sendo mais provável no passo 3 dadas as 2 primeiras palavras
- Esse processo não é otimizado para gerar a tradução
- Por outro lado, o número de combinações de palavras em uma sentença é exponencialmente grande
- Isso inviabiliza uma busca exata pela sentença ótima

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

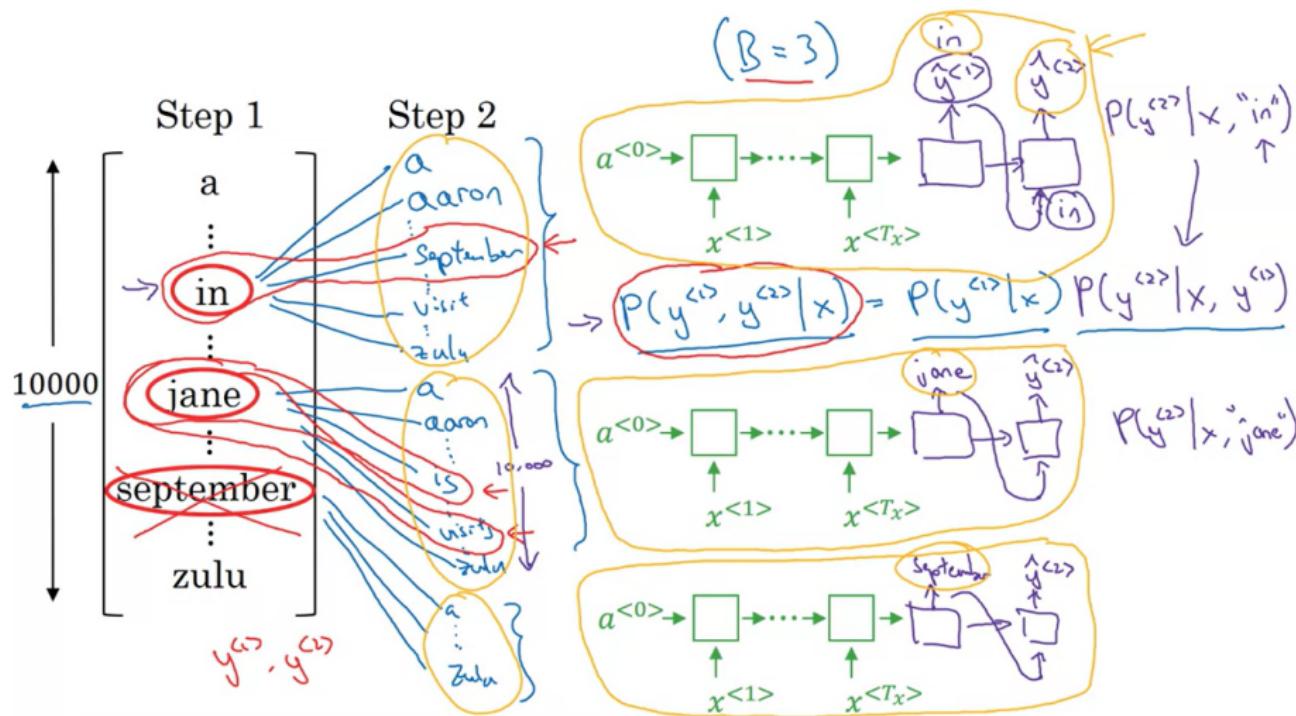
Beam Search



Beam Search

- No 1º passo olhamos para a probabilidade da 1ª palavra traduzida, dada a sentença em francês
- Porém, o *beam search* não olha apenas a palavra mais provável, mas considera várias alternativas
- Ele tem um parâmetro B , chamado largura do feixe
- Usamos aqui $B = 3$ e então ele vai considerar as 3 palavras mais prováveis

Beam Search



Beam Search

- No 2º passo, a rede vai calcular probabilidades assumindo que a 1ª palavra é cada uma das 3 escolhas do passo 1
- EX.: Na escolha de “in”, esta seria forçada a ser a saída do passo 1 e consequentemente a entrada no passo 2
- E a saída $\hat{y}^{<2>}$ seria a distribuição de probabilidades de $y^{<2>}$ dado “in”
- Estamos modelando a probabilidade do par $(y^{<1>}, y^{<2>})$, em vez da segunda palavra apenas; Essa probabilidade é o produto das probabilidades de cada palavra individual

Beam Search

- O mesmo vai ser feito para o caso da 1^a palavra ser “jane”
- Agora $y^{<1>}$ é forçado a ser “jane” e usado também como a entrada do passo 2
- A mesma coisa por fim para “september”
- Neste caso, como temos 10 mil palavras no dicionário e $B = 3$, vamos considerar 30 mil opções
- Essas 30 mil opções serão avaliadas em relação a suas probabilidades e selecionamos novamente as 3 maiores probabilidades
- As escolhas poderiam ser, p.ex., “in september”, “jane is” e “jane visits”

Beam Search

- IMPORTANTE: Se o *beam search* faz essas escolhas para o primeiro par de palavras, ele está automaticamente rejeitando a palavra “september” como 1^a palavra possível
- Temos então 2 escolhas possíveis para a 1^a palavra e 3 escolhas para o 1º par ($y^{<1>} , y^{<2>}$)
- Note também que $B = 3$ implica em instanciar 3 cópias da rede em cada passo; Mas isso permite avaliar 30 mil possibilidades sem precisar de 30 mil cópias

Beam Search

$a^{<0>} \rightarrow$ $x^{<1>} \rightarrow \dots \rightarrow x^{<T_x>} \rightarrow \hat{y}^{<3>}$

in september

α

aaron
jane
zulu

jane is

β

is
jane
zulu

jane visits

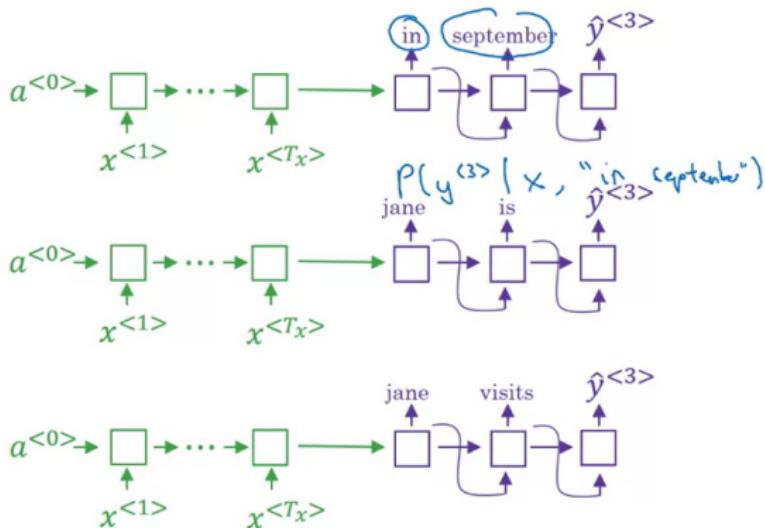
γ

visits
jane
africa
zulu

$$P(y^{<1>} , y^{<2>} | x)$$

jane visits africa in september. <EOS>

$B=1 \rightsquigarrow$ greedy search



Beam Search

- No passo 3, novamente as 3 possibilidades do passo anterior vão ser consideradas
- Para “in september”, a rede vai forçar as 2 primeiras saídas como “in september” e essas serão também as próximas entradas, como usual
- A saída no passo 3 vai ser a probabilidade de $y^{<3>}$ dado “in september”
- Mesma coisa para “jane is” e “jane visits”
- Novamente o *beam search* vai selecionar as 3 combinações mais prováveis, p.ex, “in september jane”, “jane is visiting” e “jane visits africa”

Beam Search

- E assim sucessivamente
- Esperamos que no final tenhamos uma sentença como “jane visits africa in september.” encerrada por $\langle EOS \rangle$
- NOTA: O caso $B = 1$ seria o *greedy search*

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search**
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Normalização do Comprimento

$$\begin{aligned}
 & p(y^{<1>} \dots y^{<T_y>} | x) = \frac{p(y^{<1>} | x) p(y^{<2>} | x, y^{<1>}) \dots}{p(y^{<T_y>} | x, y^{<1>} \dots, y^{<T_y-1>})} \\
 & \arg \max_y \prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>} \dots, y^{<t-1>}) \\
 & \log \left(\prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>} \dots, y^{<t-1>}) \right) \\
 & \arg \max_y \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>} \dots, y^{<t-1>}) \leftarrow \\
 & T_y = 1, 2, 3, \dots, 30. \\
 & \rightarrow \boxed{\frac{1}{T_y^\alpha} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>} \dots, y^{<t-1>})} \quad \alpha = 0.7 \quad \underline{d=1} \\
 & \qquad \qquad \qquad \underline{d=0}
 \end{aligned}$$

Normalização do Comprimento

- O beam search visa maximizar a probabilidade condicionada

$$\operatorname{argmax}_y \prod_{t=1}^{T_y} P(y^{} | x, y^{<1>}, \dots, y^{$$

- Isso é um produto de probabilidades condicionadas e cada uma delas é usualmente bem menor que 1; multiplicar esses números pode gerar *underflow*
- Em vez de maximizar o produto, tomamos o log, que transforma o produto em soma e mantém o objetivo da maximização
- Outro ponto que ajuda a melhorar o processo de tradução é a normalização

Normalização do Comprimento

- O fato de multiplicar probabilidades menores que 1 faz com que o resultado seja menor quanto mais termos eu multiplico
- Isso gera um efeito indesejado de o algoritmo de otimização preferir sentenças muito curtas
- E isso continua válido no uso de log, a diferença apenas é que neste caso o resultado é mais negativo
- A solução para isso é normalizar pelo número de palavras na sua tradução T_y
- Há também uma heurística que gera uma versão mais suave de normalização, em que se divide por T_y^α , p.ex., com $\alpha = 0.7$

Normalização do Comprimento

- Se $\alpha = 1$ temos a normalização completa e se $\alpha = 0$ não temos normalização nenhuma
- Podemos então explorar o meio entre esses extremos; α no caso é mais um hiperparâmetro a ser ajustado
- NOTA 1: Isso é uma heurística (*hack*); Não tem nenhuma justificativa teórica, mas funciona bem na prática
- NOTA 2: A função objetivo descrita costuma ser chamada de **log-likelihood normalizada**

Detalhes de Implementação

- Como escolher a largura do *beam* B ?
- B muito grande leva a resultados melhores mas é mais lento, consome mais memória
- B pequeno é mais rápido, mas o resultado é pior, já que considera menos possibilidades
- Em sistemas em produção, é comum se usar $B = 10$, sendo $B = 100$ para sistemas grandes
- Em publicações, quando o objetivo é gerar o melhor resultado possível, não é incomum que se veja $B = 1000$ até 3000

Detalhes de Implementação

- No final tudo depende muito do domínio e da aplicação
- Mas é fato que B muito grande costuma trazer ganhos marginais apenas
- De 1 (*greedy search*) para 3 ou 10, é esperado uma grande melhoria; Mas quando vamos de 1000 para 3000 esse ganho já pode ser bem pequeno
- NOTA: Algoritmos como BFS (*Breadth First Search*) ou DFS (*Depth First Search*) seriam alternativas exatas ao *beam search*; Mas o *beam search* é muito mais rápido

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Análise de Erros

- Ajuda saber em qual parte do modelo focar mais
- O *beam search* é uma heurística e portanto nem sempre ele retorna a melhor sentença; O que fazer quando ele comete um erro?
- Vamos descobrir se o problema está no *beam search* ou na rede recorrente e assim definir onde devemos gastar mais tempo

Análise de Erros

Jane visite l'Afrique en septembre.

→ RNN

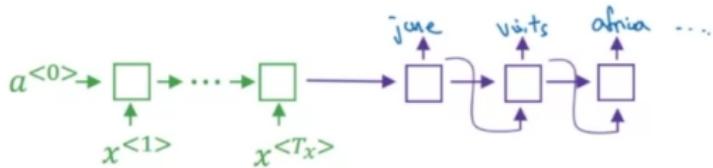
→ Beam Search



Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. (\hat{y}) ←

$$\text{RNN computes } P(y^*|x) \stackrel{>}{\leq} P(\hat{y}|x)$$



Análise de Erros

- Continuamos com o exemplo “Jane visite l’Afrique em septembre”
- Suponha que no *dev set* um humano ofereceu a tradução “Jane visits Africa in september”; Vamos chamá-la de y^*
- E suponha que ao rodarmos o *beam search* no modelo treinado, temos “Jane visited Africa in september”
- É uma tradução pior do que a do humano, já que muda o significado; Vamos chamá-la de \hat{y}
- O modelo tem dois componentes principais: a RNN (*encoder/decoder*) e o *beam search* (com alguma largura de *beam B*)
- Seria interessante atribuir essa tradução errada como sendo causada pela RNN ou pelo *beam search*

Análise de Erros

- Vimos na análise de erros que era sempre tentador coletar mais dados de treino
- Aqui, do mesmo modo, é tentador aumentar o B , embora aumente o custo computacional
- O problema é que, assim como aumentar o treino por si só pode não ajudar tanto, aumentar B também pode não levar ao resultado desejado
- A RNN treinada calcula $P(y|x)$; Podemos então calcular $P(y^*|x)$ e $P(\hat{y}|x)$
- Dependendo de qual probabilidade é maior, podemos associar o responsável principal pelo problema, a RNN ou o *beam search*

Análise de Erros

Human: Jane visits Africa in September. (y^*)

$$P(y^*|x)$$

Algorithm: Jane visited Africa last September. (\hat{y})

$$P(\hat{y}|x)$$

Case 1: $P(y^*|x) > P(\hat{y}|x)$ \leftarrow

$$\arg \max_y P(y|x)$$

Beam search chose \hat{y} . But y^* attains higher $P(y|x)$.

Conclusion: Beam search is at fault.

Case 2: $P(y^*|x) \leq P(\hat{y}|x)$ \leftarrow

y^* is a better translation than \hat{y} . But RNN predicted $P(y^*|x) \leq P(\hat{y}|x)$.

Conclusion: RNN model is at fault.

Análise de Erros

- Caso 1:

$$P(y^*|x) > P(\hat{y}|x)$$

O *beam search* escolheu $P(\hat{y}|x)$ através de $\text{argmax}_y P(y|x)$.
Mas y^* tem $P(y|x)$ maior.

Portanto, o *beam search* está com problema neste caso, já que seu único papel era maximizar $P(y|x)$.

- Caso 2:

$$P(y^*|x) \leq P(\hat{y}|x)$$

Aqui y^* é uma tradução melhor que \hat{y} , mas a RNN previu $P(y^*|x) \leq P(\hat{y}|x)$.

Portanto, a RNN está com problema aqui.

Análise de Erros

- Existem ainda algumas sutilezas como o fato de que, se estiver usando normalização do comprimento, deve-se usar em vez das probabilidades, a função objetivo normalizada

Análise de Erros

Human	Algorithm	$P(y^* x)$	$P(\hat{y} x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	2×10^{-10}	1×10^{-10}	B
...	---	—	—	R
...	---	—	—	Q
				R R R ⋮

Figures out what fraction of errors are “due to” beam search vs. RNN model

Análise de Erros

- A análise de erro começa por analisar sentenças com tradução ruim no *development set*
- Observamos vários exemplos e dependendo das probabilidades atribuímos o problema à rede ou ao *beam search*
- E então observamos a proporção de erros atribuídos a cada componente

Análise de Erros

- Se a proporção de problemas for alta no *beam search*, pode ser interessante mexer no B
- Se o problema for na RNN, podemos usar mais regularização, mais dados de treino, uma arquitetura diferente, etc.
- Em geral, a análise de erros é bastante útil em cenários com algoritmos de otimização aproximados como o *beam search*

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Bleu Score³

- Um desafio na tradução automática é a existência de múltiplas traduções possíveis e igualmente boas
- Isso é diferente, por exemplo, do reconhecimento de imagens, em que há uma resposta correta apenas
- Como medir então a qualidade de uma tradução?
- A medida mais usada é o **Bleu score**

³Papineni et. al., 2002. Bleu: A method for automatic evaluation of machine translation

Bleu score

French: Le chat est sur le tapis.

→ Reference 1: The cat is on the mat.

→ Reference 2: There is a cat on the mat.

→ MT output: the the the the the the the

$$\text{Precision: } \frac{7}{7}$$

Modified precision:

$$\frac{2}{7} \leftarrow \begin{array}{l} \text{Count}_{clip} ("the") \\ \text{Count} ("the") \end{array}$$

Bleu
bilingual evaluation understudy

Bleu score

- Suponha a sentença em francês “Le chat est sur le tapis.” e uma tradução humana de referência “The cat is on the mat.”
- Mas há outras boas traduções como “There is a cat on the mat.”
- Para uma tradução próxima da humana, o *Bleu score* vai ser alto
- NOTA: Bleu é a sigla de *Bilingual Evaluation Understudy*
- A palavra “understudy” no teatro se refere a alguém que aprende o papel de um ator mais sênior, podendo se necessário substituí-lo
- A ideia é que o *Bleu score* possa assumir o papel de um humano na avaliação da tradução

Bleu score

- Os autores são Kishore Papineni, Salim Roukos, Todd Ward e Wei-Jing Zhu; É um texto bem tranquilo de ler
- Supondo agora que a saída da *machine translation* (MT) seja muito ruim, como “the the the the the the.”
- A primeira medida que vamos definir é a **Precisão**, que é a fração de palavras na tradução automática que aparecem nas traduções de referência.
- No caso, todas as palavras de MT (“the”) aparecem; Portanto, nossa Precisão seria 7/7

Bleu score

- Definimos também a **Precisão Modificada**, em que temos o conceito de crédito para cada palavra, que corresponde ao número máximo de vezes que a palavra aparece em uma sentença de referência
- No caso, “the” aparece duas vezes na primeira referência
- Nossa medida de Precisão Modificada seria então 2/7, já que o numerador é limitado (*clip*) pelo crédito da palavra
- Até agora olhamos para palavras isoladas (unigramas); O *Bleu score* também olha para pares de palavras (bigramas)

Bleu score

Example: Reference 1: The cat is on the mat. ↪

Reference 2: There is a cat on the mat. ↪

MT output: The cat the cat on the mat. ↪

	Count	Count clip	
the cat	2 ↪	1 ↪	
cat the	1 ↪	0	
cat on	1 ↪	1 ↪	
on the	1 ↪	1 ↪	
the mat	1 ↪	1 ↪	
			<u>4</u> <u>6</u>

Bleu score

- Suponha que a MT agora seja “The cat the cat on the mat.”; Não é ainda uma boa tradução, mas melhor que antes
- Temos então os bigramas, ignorando maiúscula/minúscula:
the cat
cat the
cat on
on the
the mat
- Fazendo a contagem, temos que “the cat” aparece 2 vezes na MT e os demais

Bleu score

- Agora calculamos a contagem limitada pelo crédito, que conta quantas vezes no máximo o bigrama aparece em pelo menos uma das referências
- No caso temos 0 para “cat the” e 1 para os demais
- A Precisão Modificada será então a razão entre o número de bigramas e o número limitado (soma de $count_{clip}$): $4/6$

Bleu score

Example: Reference 1: The cat is on the mat.

$$P_1, P_2, = 1.0$$

Reference 2: There is a cat on the mat.

→ MT output: The cat the cat on the mat. (\hat{y})

$$P_1 = \frac{\sum_{\text{unigrams} \in \hat{y}} \text{Count}_{clip}(\text{unigram})}{\sum_{\text{unigram} \in \hat{y}} \text{Count}(\text{unigram})}$$

↑
Unigram

$$P_n = \frac{\sum_{n\text{-grams} \in \hat{y}} \text{Count}_{clip}(n\text{-gram})}{\sum_{n\text{-gram} \in \hat{y}} \text{Count}(n\text{-gram})}$$

↑
n-gram

Bleu score

- Definimos p_1 como a Precisão Modificada sobre unigramas, assim como p_n para n-gramas
- Temos então uma medida de sobreposição entre a tradução automática e as referências humanas
- NOTA: Se MT for exatamente igual à Referência 1 ou à 2, então p_1 , p_2 , etc. serão todos iguais a 1.0
- Embora esse valor também possa ser atingido sem ser exatamente igual a uma referência, mas sim meio que combinando as duas referências

Bleu score

p_n = Bleu score on n-grams only

P_1, P_2, P_3, P_4

Combined Bleu score: $\text{BP} \exp\left(\frac{1}{4} \sum_{n=1}^4 p_n\right)$

BP = brevity penalty

$$\text{BP} = \begin{cases} 1 & \text{if } \underline{\text{MT_output_length}} > \underline{\text{reference_output_length}} \\ \exp(1 - \text{reference_output_length}/\text{MT_output_length}) & \text{otherwise} \end{cases}$$

Bleu score

- p_n seria o *Bleu score* sobre n-gramas apenas
- O que temos então é o *Bleu score* combinado, em que se usa por convenção a exponencial da média de p_1 , p_2 , p_3 e p_4 , multiplicado por BP (*brevity penalty*)
- BP penaliza traduções muito curtas, já que neste caso seria mais fácil ter palavras em comum com a referência

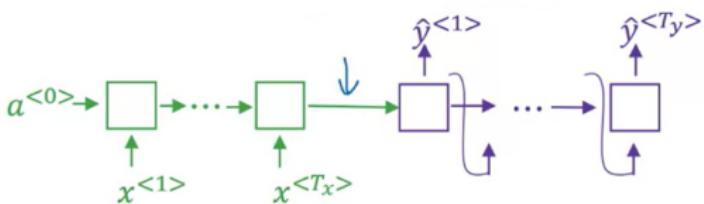
Bleu score

- O *Bleu score* foi muito importante porque, como vimos, ter um único número real que avalia o desempenho de um modelo é muito útil
- É usado em outros contextos que geram textos também, como nas legendas automáticas
- Já em áreas como o reconhecimento de fala, ele não costuma ser usado porque lá existe uma única resposta exata esperada

Outline

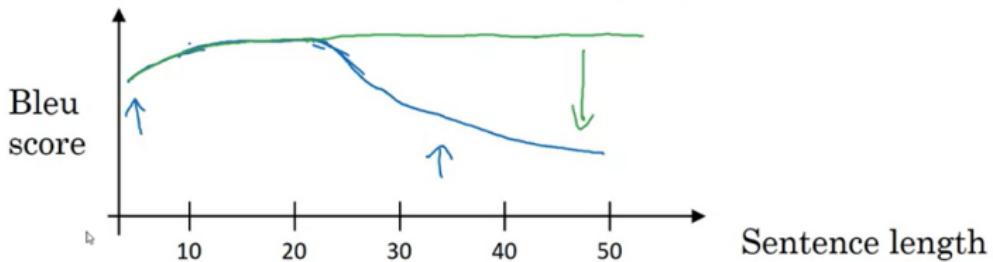
- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção**
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Intuição⁴



Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



⁴Bahdanau et al., 2014. Neural machine translation by jointly learning to align and translate

Intuição

- Modificação da ideia de *encoder/decoder* que melhora muito o algoritmo
- Uma das ideias mais revolucionárias do *deep learning*!
- Até agora o modelo lia uma sentença, possivelmente muito longa, memorizava e transmitia através da ativação que entra no *decoder*
- Mas um ser humano não traduz uma sentença desse jeito!
- Ele lê uma parte da sentença e gera algumas palavras, mais uma parte e gera mais palavras e assim por diante

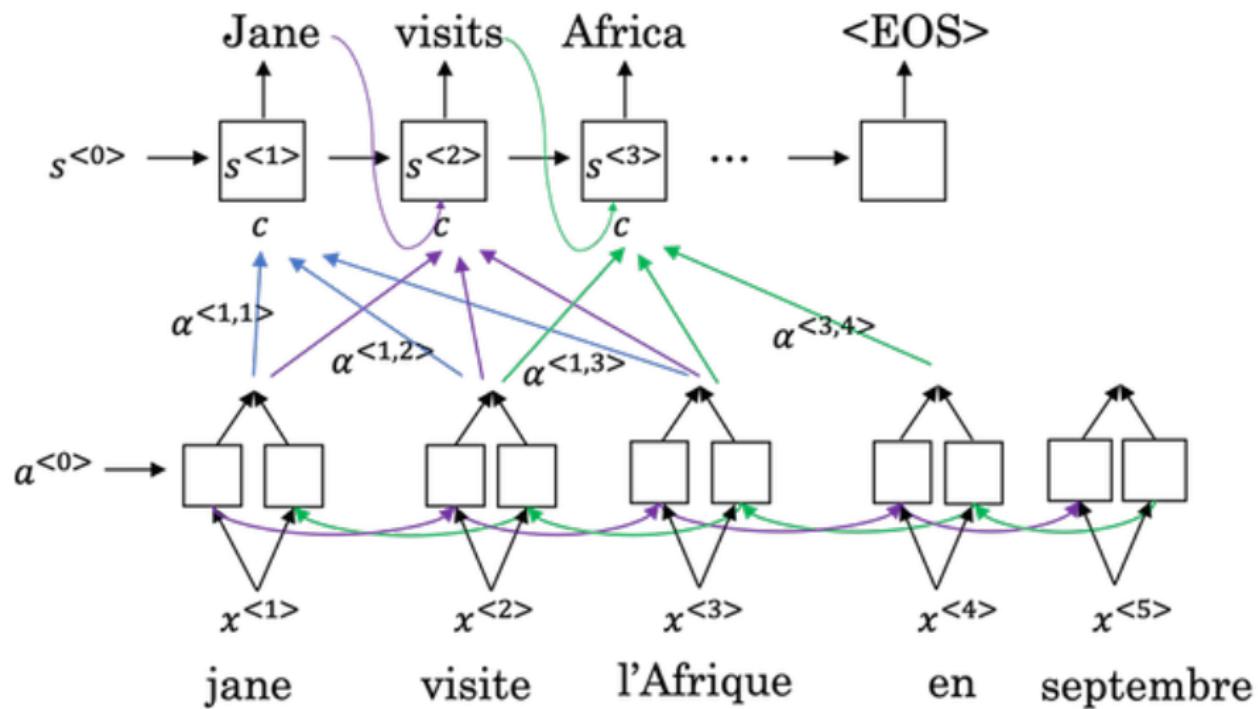
Intuição

- O modelo *encoder/decoder* funciona para sentenças curtas, mas não para as mais longas
- O *Bleu score* se parece com a curva azul na figura
- O modelo de atenção traduz por partes e assim o *Bleu score* se mantém mais alto todo o tempo (curva verde)
- A distância entre a curva verde e a azul mede a capacidade de memorizar longas sentenças e esse *gap* grande não é interessante

Intuição

- NOTA: Os autores são Dimitri, Bahdanau, Camcrun Cho e Yoshe Bengio
- Embora tenha sido desenvolvido para tradução, a ideia se espalhou para muitas áreas
- Foi um trabalho realmente muito influente e seminal na literatura!

Intuição



Intuição

- Temos a nossa sentença usual “jane visite l'Afrique en septembre”
- Construímos uma RNN bidirecional que extrai *features* das palavras da sentença em francês
- Removemos as saídas y da RNN porque não vamos fazer uma tradução palavra-a-palavra
- Essa RNN, para cada palavra, calcula um conjunto de *features* e possivelmente das palavras em volta também
- Para a tradução usamos outra RNN.
- Chamaremos os estados escondidos agora de s

Intuição

- Esperamos que no 1º passo dessa RNN de tradução a palavra gerada seja “jane”
- PERGUNTA: Na geração dessa palavra de saída, para qual parte da sentença em francês o modelo deve olhar?
- É de se supor que precise olhar mais para a primeira palavra e talvez algumas mais em volta, mas nada no final da sentença
- O modelo de atenção calcula um conjunto de **pesos de atenção**
- Chamamos de $\alpha^{<1,1>}$ que nos diz o quanto de atenção devemos prestar à 1ª palavra em francês para gerar a 1ª palavra da tradução

Intuição

- Do mesmo modo, $\alpha^{<1,2>}$ nos diz quando te atenção prestar à 2^a palavra na obtenção da 1^a palavra da tradução (que esperamos ser “jane”)
- E assim por diante
- Isso nos dá o contexto c ao qual a 1^a unidade RNN na tradução deve prestar atenção
- Esse é o 1º passo da RNN
- No 2º passo, temos o novo estado escondido $s^{<2>}$ e os pesos $\alpha^{<2,1>} , \alpha^{<2,2>} ,$ etc., que mostram o quanto se deve prestar de atenção para gerar a 2^a palavra (que esperamos ser “visits”)

Intuição

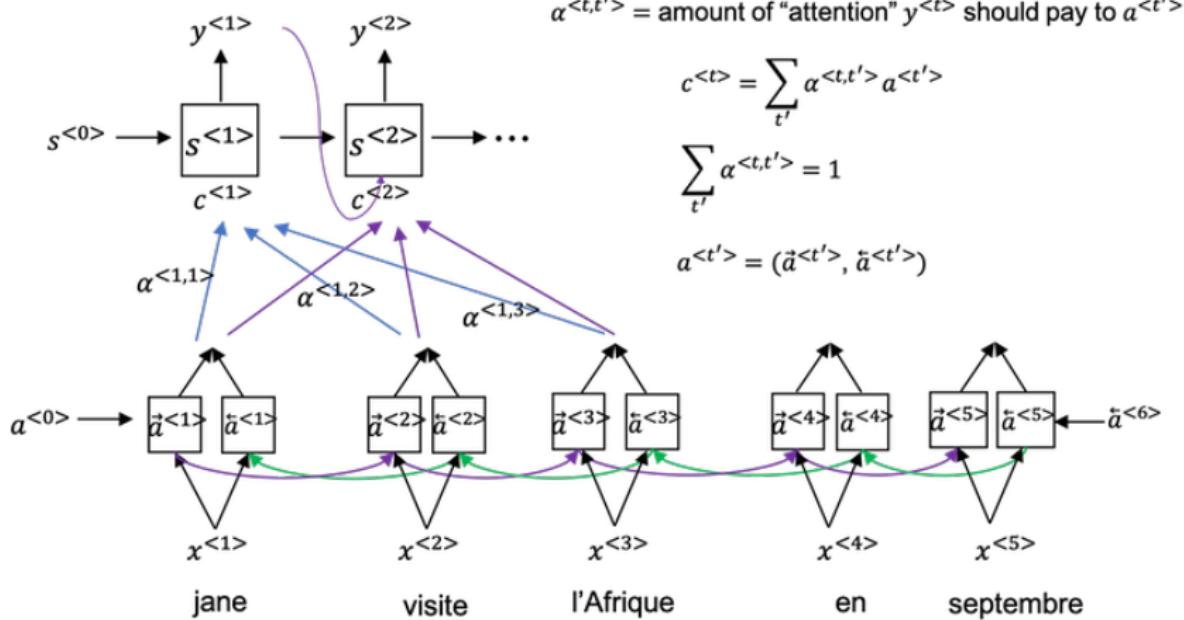
- Como vimos, a 1^a palavra gerada é usada como entrada para gerar a 2^a e se junta então com o contexto c
- No 3º passo ocorre algo similar e assim prosseguimos
- O objetivo do vetor c é, por exemplo, na geração da 3^a palavra focar na 3^a palavra e nas que estão em volta da sentença original
- Veremos em breve as fórmulas exatas para calcular esses pesos
- Veremos, por exemplo, de $\alpha^{<3,t>}$ depende das ativações $a^{<t>}$ nas duas direções e do estado escondido $s^{<2>}$
- E assim como antes a sentença traduzida vai sendo gerada palavra por palavra até encontrar um $<EOS>$

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização**
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Formalização

Attention model



Formalização

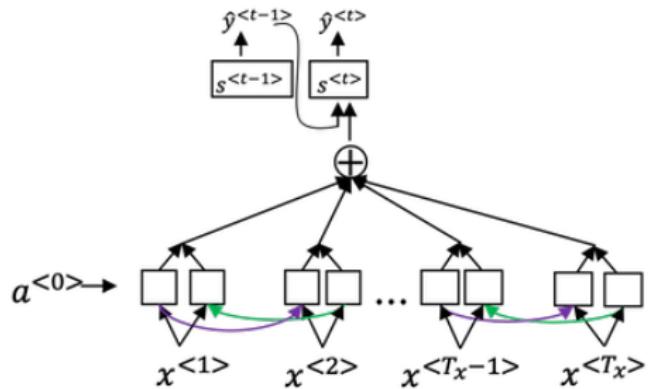
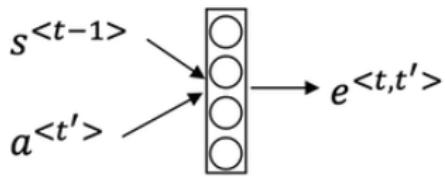
- Vamos usar a mesma sentença de antes e uma rede bidirecional (Blocos GRU e principalmente LSTM são comuns aqui)
- Na rede bidirecional temos uma ativação $a^{<0>}$ de entrada na rede *forward* e uma $a^{<6>}$ de entrada na rede *backward* que contém apenas zeros
- Chamamos a ativação concatenada da parte *forward* e *backward* no tempo t' da sentença em francês de $a^{<t'>}$
- Esse é o *feature vector* da sentença de entrada naquele passo de tempo
- E temos agora a rede que gera a tradução, que é unidirecional

Formalização

- Em cada passo, ela recebe a ativação anterior e o contexto c
- O contexto é a soma dos pesos, que no exemplo do passo 1 seriam $\alpha^{<1,t'>}$ multiplicados pelas ativações, no caso, $a^{<t'>}$
- Temos ainda a restrição de que os pesos α no passo de tempo devem somar 1
- Definimos $\alpha^{<t,t'>}$ como a quantidade de “atenção” que $y^{<t>}$ deve prestar a $a^{<t'>}$
- E o processo se repete para os demais passos de tempo

Cálculo dos Pesos

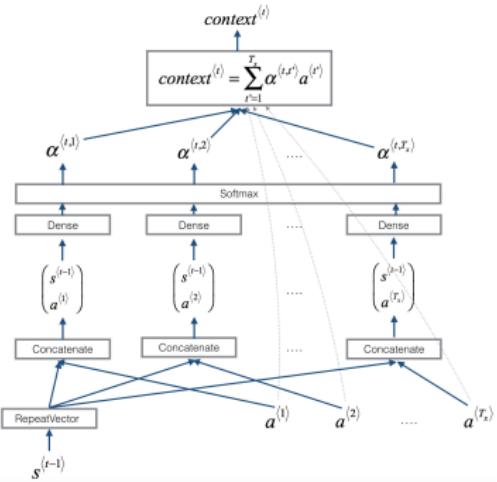
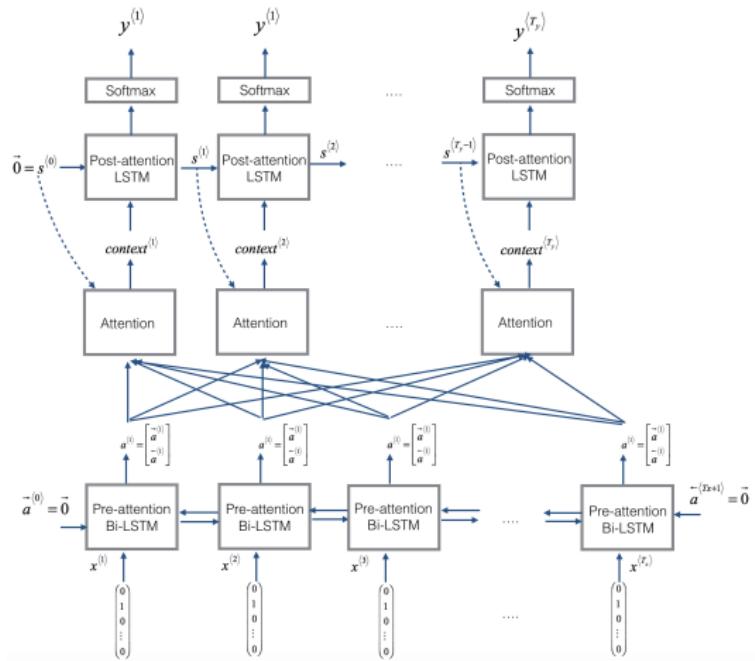
$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$



Cálculo dos Pesos

- Os pesos são calculados por uma softmax aplicada sobre $e^{}$
- $e^{}$ é uma rede neural de uma camada que recebe como entrada $s^{}$ e $a^{}$
- Faz sentido que a quantidade de atenção dependa do *featurevector* em t' e do estado anterior na tradução
- Porém, como essa relação não pode ser estabelecida *a priori*, cria-se uma rede neural para aprendê-la
- A softmax também garante que os pesos somem 1 como desejado

Arquitetura Geral



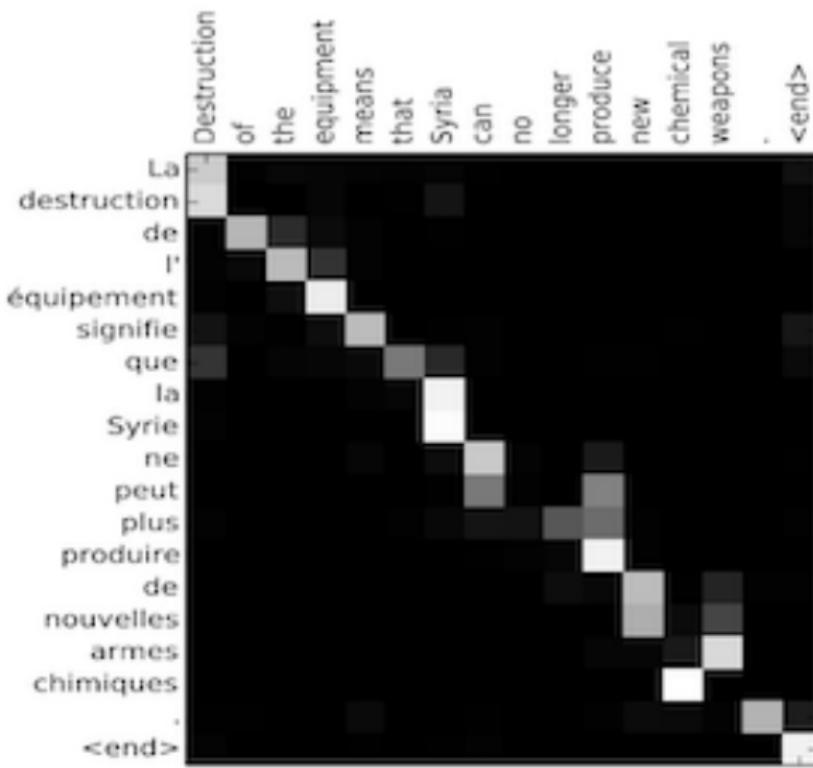
Problemas e Aplicações

- Uma desvantagem desse algoritmo é que ele roda em tempo quadrático
- Se as sentenças original e traduzida tiverem comprimento T_x e T_y , o número de parâmetros de atenção será $T_x T_y$
- Na tradução automática, normalmente as sentenças não são tão longas; Mas há pesquisas tentando reduzir esse custo
- Ideia também foi aplicada fora de tradução, por exemplo, na geração de legendas em
[Xu et al., 2015. Show, attend and tell: Neural image caption generation with visual attention]
de Kevin Chu, Jimmy Barr, Ryan Kiros, Kelvin Shaw, Aaron Korver, Russell Zarkutnov, Virta Zemo e Andrew Benjo

Problemas e Aplicações

- A arquitetura no caso é bem parecida e a ideia é olhar apenas para partes da imagem enquanto vai gerando a legenda
- Outra coisa legal é visualizar os pesos de atenção, normalmente usando um mapa de cores para a magnitude do peso

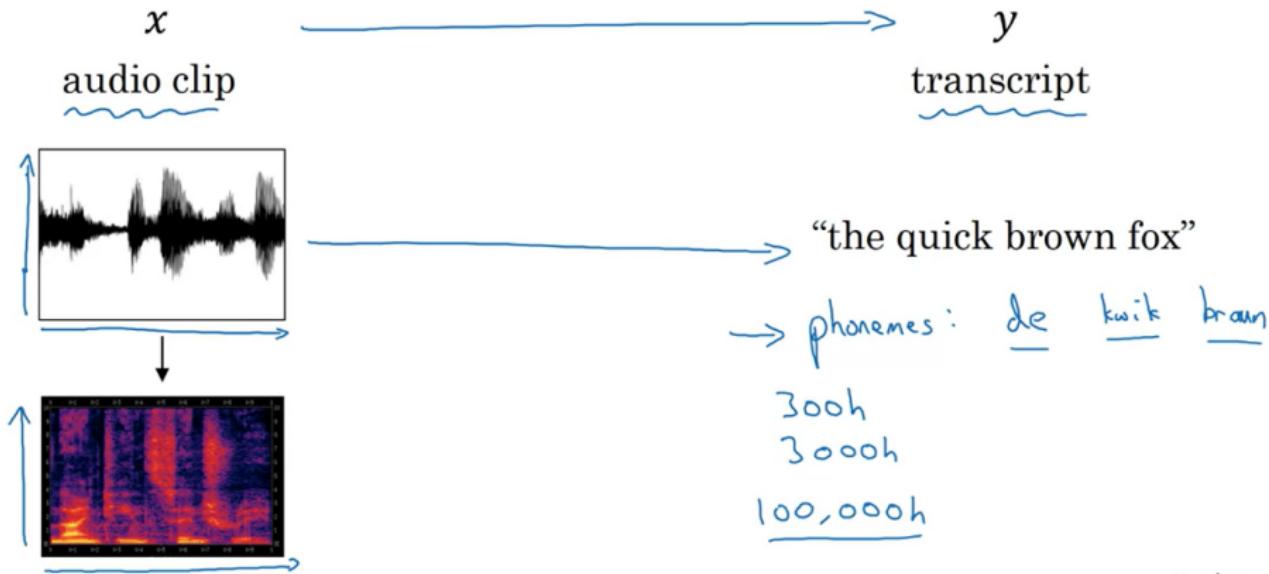
Visualização



Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

Dados de Áudio



Dados de Áudio

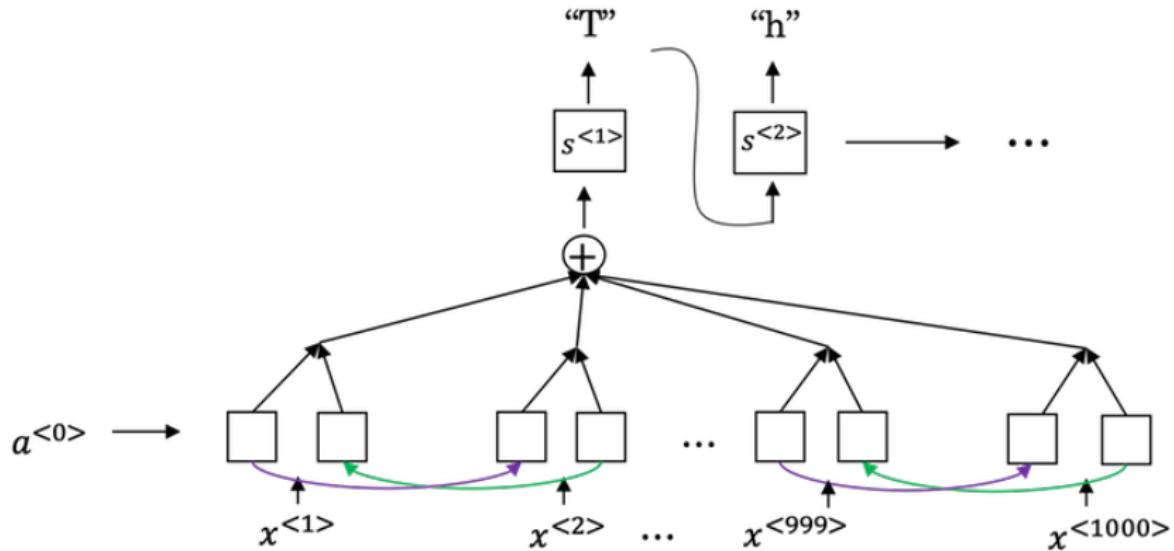
- Objetivo no reconhecimento de áudio é receber um clipe x e retornar uma transcrição em texto y
- No clipe de áudio o eixo horizontal é o tempo e o vertical são mudanças minúsculas na pressão do ar
- Porém, mesmo a audição humana não processa o áudio em sua forma bruta, mas sim medindo a intensidade de diferentes frequências
- E assim um pré-processamento comum para áudio é gerar um espectrograma
- Nessa representação, o eixo horizontal é o tempo, o vertical é a frequência e a intensidade das cores corresponde à quantidade de energia em cada frequência

Dados de Áudio

- O espectrograma é chamado também de *false-back output*
- No passado o reconhecimento de fala envolvia a quebra de sentenças em fonemas, algo que era totalmente “*hand-engineered*” e envolvia conhecimentos de linguistas
- Com a abordagem ponta-a-ponta do *deep learning* isso não é mais necessário
- Uma das coisas que possibilitou isso foi o uso de conjuntos enormes de dados
- Na pesquisa acadêmica um conjunto com 3000 horas de áudio é considerado razoável, enquanto aplicações comerciais envolvem 10 mil até 100 mil horas

Dados de Áudio

Attention model for speech recognition

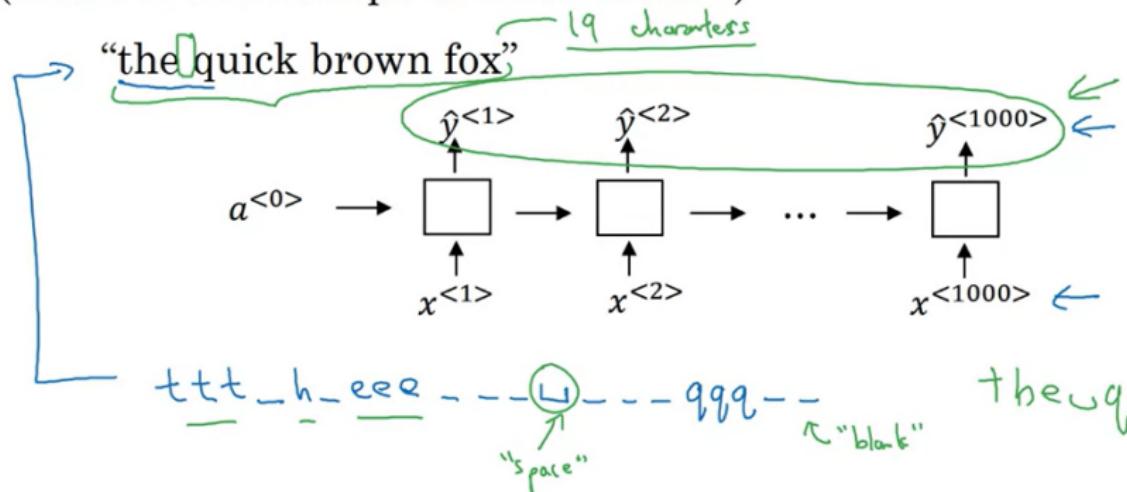


Dados de Áudio

- Mas como construir um sistema de reconhecimento de fala?
- Uma possibilidade é um modelo de atenção que recebe *time frames* do áudio e vai gerando a transcrição

CTC Cost⁵

(Connectionist temporal classification)



Basic rule: collapse repeated characters not separated by “blank”

⁵Graves et al., 2016. Connectionist Temporal Classification: Labeling unsegmented sequence data with recurrent neural networks

CTC Cost

- Um método que funciona bem é o CTC *cost* (*Connectionist temporal classification*) para reconhecimento de fala
- AUTORES: Alex Graves, Santiago Fernandes, Faustino Gomez e Jürgen Schmidhuber
- Ilustramos com uma RNN unidirecional, mas normalmente isso é implementando com uma GRU ou LSTM bidirecional e usualmente profunda
- No reconhecimento de fala normalmente o número de passos de tempo da entrada é muito maior que o da saída

CTC Cost

- A CTC *cost function* alinha entrada e saída repetindo o caractere, além de um caractere especial *blank* (representado na figura por um sublinhado), assim como um caractere especial *space*
- A regra básica é colapsar caracteres repetidos não separados por “*blank*”
- Tanto o CTC quanto modelos de atenção funcionam bem nessa aplicação
- Porém, sistemas efetivos de reconhecimento de fala precisam de uma grande quantidade de dados
- Já uma tarefa bem mais fácil e que não requer tanto dado é o reconhecimento de palavras-gatilho (palavras-chave)

Outline

- 1 Modelos Sequência-para-Sequência
- 2 Tradução e Sentença Mais Provável
- 3 Beam Search
- 4 Refinamentos do Beam Search
- 5 Análise de Erros no Beam Search
- 6 Bleu Score
- 7 Modelos de Atenção
- 8 Modelo de Atenção - Formalização
- 9 Reconhecimento de Fala
- 10 Reconhecimento de Palavras-Gatilho

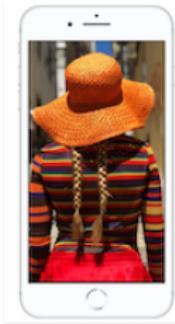
Palavras-Gatilho



Amazon Echo
(Alexa)



Baidu DuerOS
(xiaodunihao)



Apple Siri
(Hey Siri)

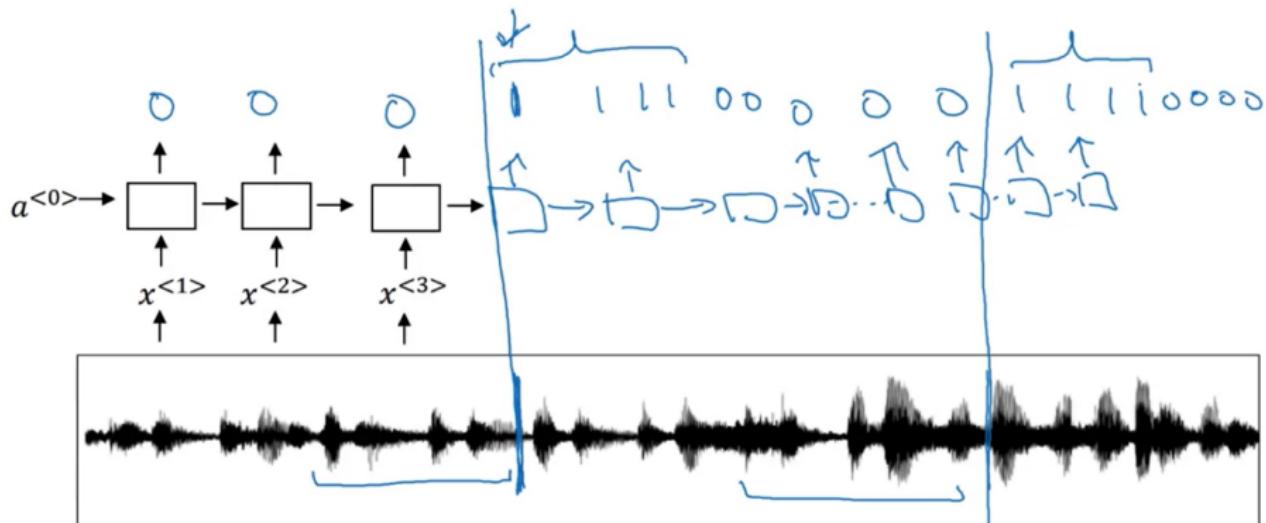


Google Home
(Okay Google)

Palavras-Gatilho

- A detecção de palavras-gatilho é a aplicação de aparelhos que podem ser “acordados” com certas palavras
- Projetos caseiros como acender/apagar uma lâmpada com certas palavras também são possíveis
- A área ainda está em evolução, mas um algoritmo possível consiste em processar o clipe (possivelmente com *features* de espectrograma) por uma RNN
- E em cada passo de tempo (único) no treinamento em que a palavra-gatilho começa a ser falada, temos $y = 1$, senão $y = 0$

Palavras-Gatilho



Palavras-Gatilho

- Embora isso funcione razoavelmente, uma desvantagem é que o conjunto de treino fica muito desbalanceado, com muito mais 0s do que 1s
- Um *hack* para isso é repetir 1 a partir do ponto em que a palavra é falada por vários passos de tempo (por exemplo um número fixo de passos)