Redes Neurais Recorrentes

Aplicadas para o Processamento de Linguagem Natural

Agenda

1) Apresentações
2) Introdução
3) Topologias
    a) GRU
    b) LSTM
4) Exemplos de estruturas para NLP
5) Conclusões

Apresentações

Daniel Dominguete

Head de P&D da Omnilogic (www.omnilogic.ai)
Founder Cognas (www.cognas.ai)
Founder Ringabell (www.ringabell.com.br)

Experiências anteriores:
Siemens/Chemtech, Parex Engenharia, DDMX, Ottimah Process Improvements, Faculdade Pitágoras.

Formação:
Graduação em Eng. Elétrica/UFMG-2003/1
Mestre em IA/UFMG 2004/2
MBA em Administração de Empresas/FGV -2007/2
Especialista em Eng. Industrial/PUC-MG 2014/2

Música: rockforthband

ministerio.natividad

OMNILOGIC

POWER THROUGH CONNECTIONS

Empresa de **engenharia cognitiva** que oferece soluções para a **convergência entre máquinas e humanos**, ao fazer com que o digital seja o mais orgânico e natural possível em um processo de **Humanização da Tecnologia**.

A partir do entendimento semântico e comportamental é possível desenvolver empatia para a **construção de conexões efetivas**.

Estas conexões aceleram o crescimento de empresas e indivíduos ao ampliar suas capacidades de **troca de valor.**

AVON the company for women · B2W DIGITAL · bradesco · buscapé company · CVC

LATAM AIRLINES · Magalu · SEPHORA

Viagens · T4f TIME FOR FUN · VALE · Walmart

almundo the travellers' community · + soma GRUPO DE MODA · dr.consulta

PRODUCT CLOUD · **INTELLIGENT STORE** · **CUSTOMER CLOUD** · **PERSONAL ASSISTANT**

Big Data & Data Science · Data Extraction · Knowledge Science · Semantics & Perceptron · Cognitive Computing

Principais Parceiros: Microsoft · VTEX · SCALE UP ENDEAVOR · UFMG · UFLA

cognas
cognitive and analytical solutions

Hub de Conhecimento em Data Science

Blog | Mentoria | Eventos | Serviços

Contatos:
Site: www.cognas.ai
Instagram: @cognas.ai
Linkedin: /cognas
Medium: /cognas
E-mail: cognas@cognas.ai

SCAN ME

RingaBell

Mentoria Digital para Empreendedores

Hub de Conhecimento em Transformação Digital

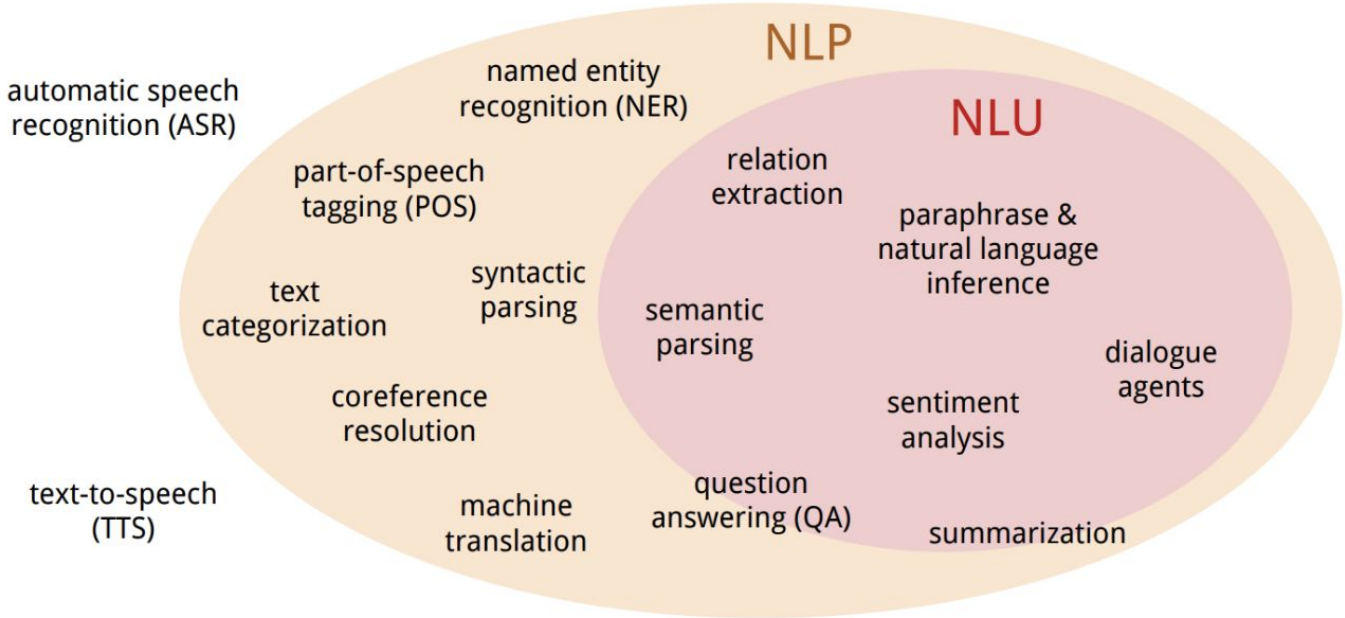
Blog | Mentoria | Eventos | Serviços

Contatos:
Site: www.ringabell.com.br
Instagram: @ringabelloficial
Linkedin: /ringabell
Medium: /ringabell-business
E-mail: contato@ringabell.com.br

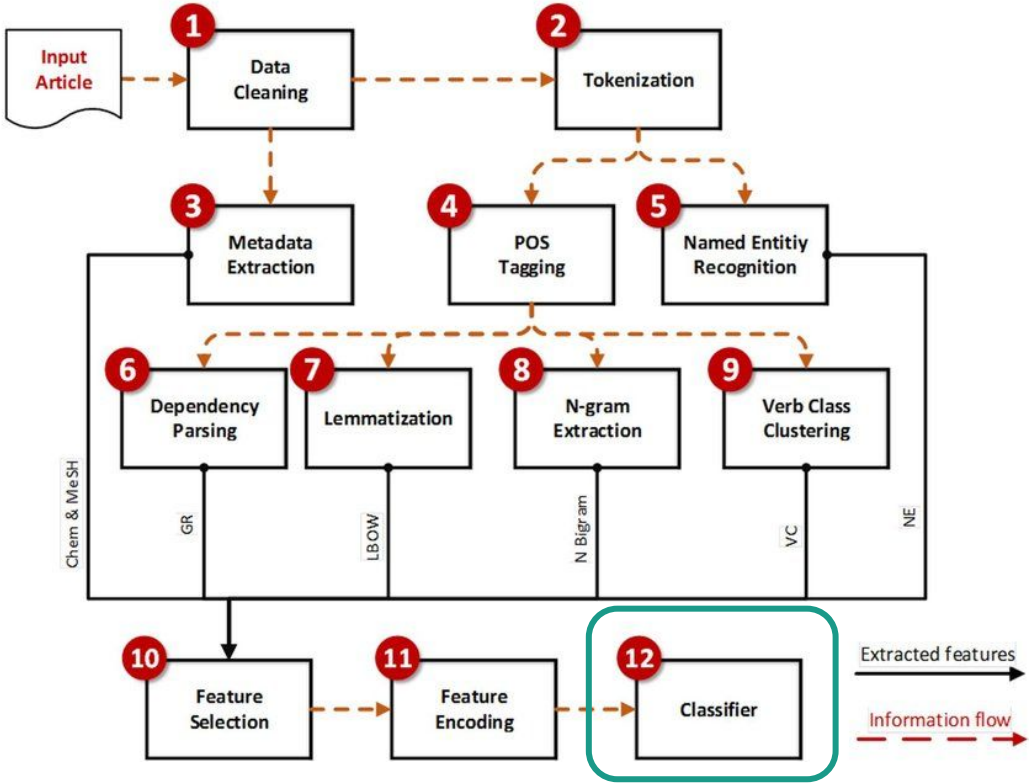
SCAN ME

Introdução

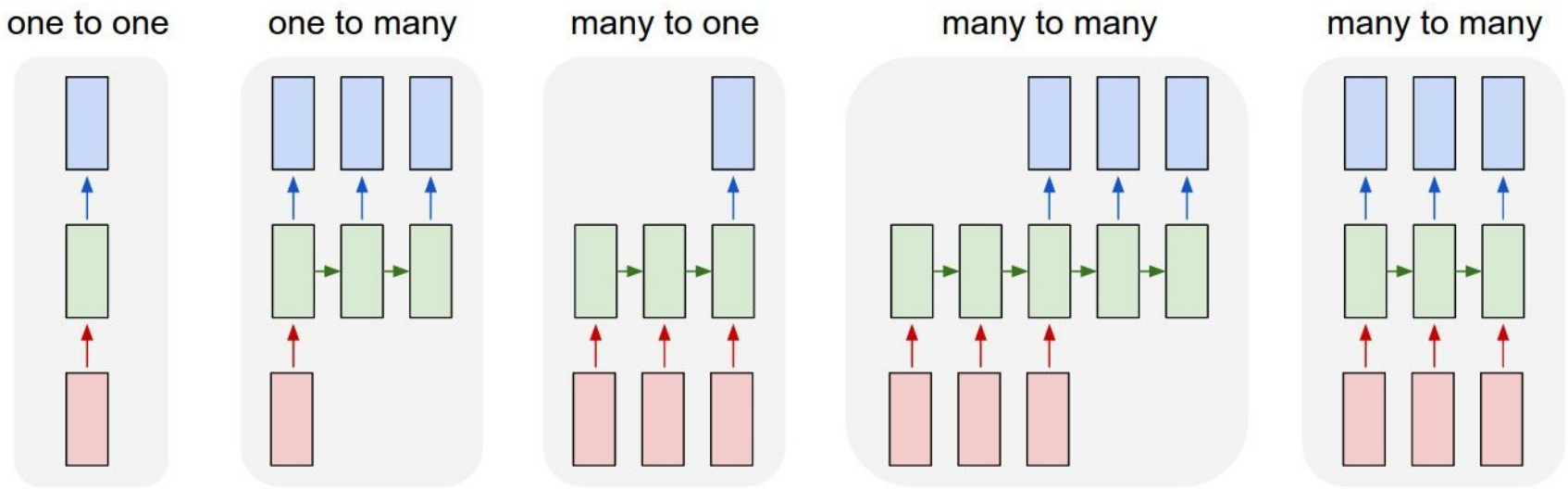
Processamento de Linguagem Natural



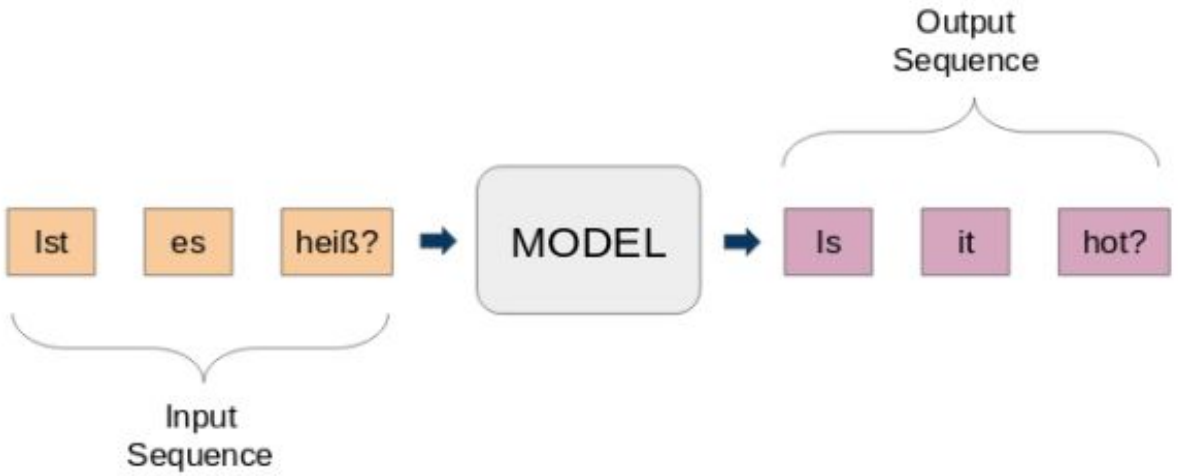
Exemplo de Pipeline NLP



Casos de Uso de Modelos Neurais

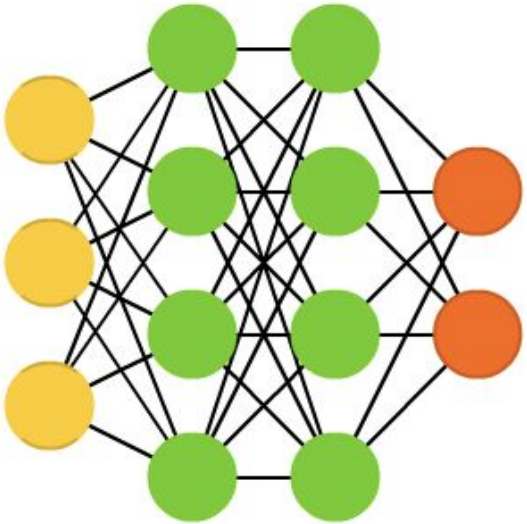


Exemplo de Tradução Automática

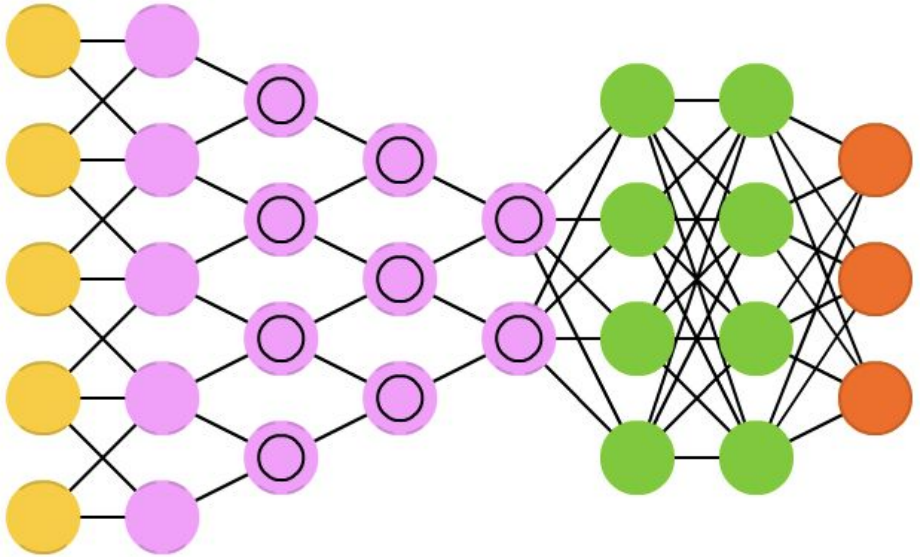


Redes Neurais Não-Recorrentes

Deep Feed Forward (DFF)



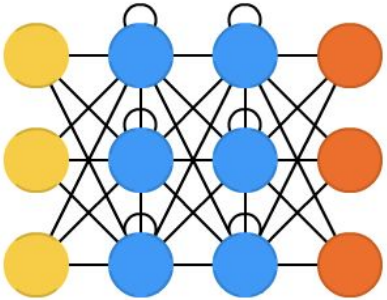
Deep Convolutional Network (DCN)



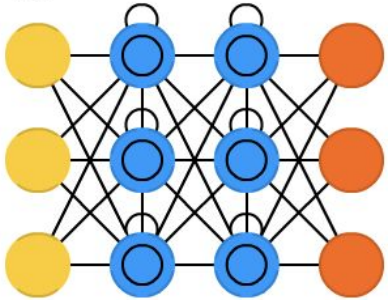
Não existe uma dependência de tempo, ou seja, não existe processamento sequencial. Além de trabalharem com vetores de tamanho sequencial fixo na entrada e saída.

Redes Neurais Recorrentes

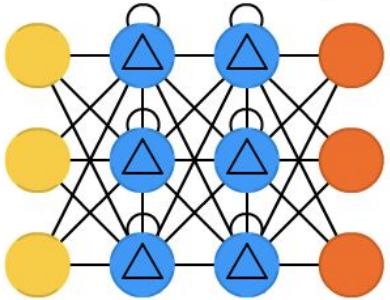
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)

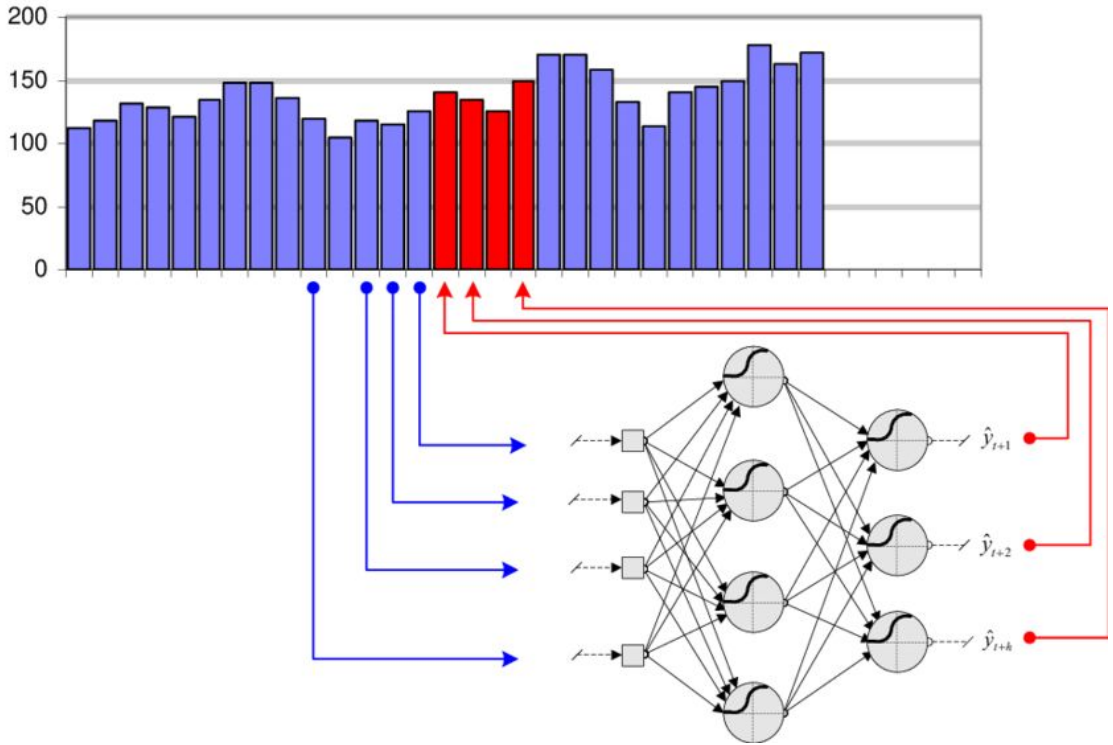


Gated Recurrent Unit (GRU)



Existe retroalimentação da informação internamente, levando em consideração a sequência das informações. Permitem trabalhar com vetores de tamanhos variados sequenciais tanto na entrada quanto na saída.

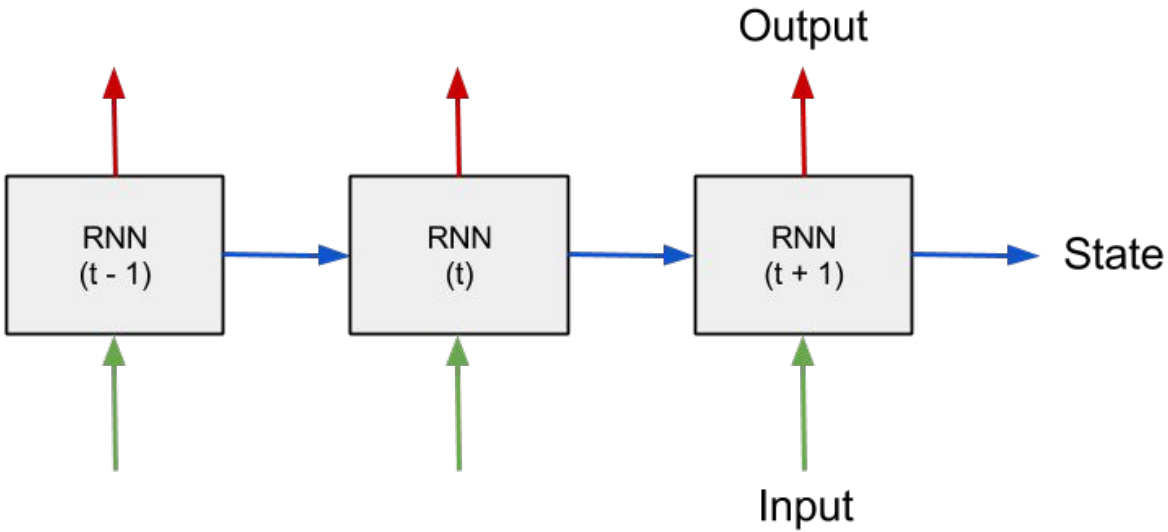
Séries Temporais com Neural Networks



Método de slide windows:

- Um neurônio de entrada para cada instante de tempo das amostras de entrada.
- Um neurônio de saída para cada instante de tempo a ser predito.

Séries Temporais com Recurrent Neural Networks



Método Time Steps:

- A relação temporal é representada pelos estados internos da rede neural.
- A quantidade de neurônios de entrada depende somente da dimensionalidade do vetor de entrada.
- A camada de entrada é avaliada para cada passo da sequência, e não somente o neurônio representativo do passo. Não existe o conceito de um neurônio para cada passo.

Diferença da Modelagem

Redes Não Recorrentes: Assumimos uma “independência” das variáveis

$$X_{t+1} = f(X_t, X_{t-1}, X_{t-2}, \ldots, X_{t-n})$$

Redes Recorrentes: Assumimos variáveis interdependentes.

$$X_{t+1} = f(X_t, f(X_{t-1}, f(X_{t-2}, \ldots, f(X_{t-n}))))$$

Séries Temporais e NLP

"É preciso deixar essa história para boi dormir de lado".

“1 2 3 4 5 6 7 8 9 10”

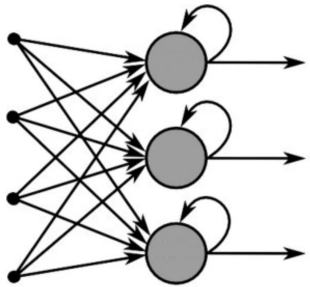
"É preciso deixar de lado essa história para boi dormir".

“1 2 3 9 10 4 5 6 7 8”

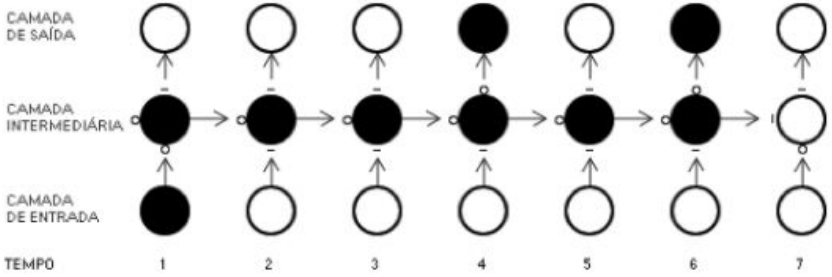
A ordem temporal das informações é fator relevante na construção do conhecimento.

Redes Neurais Recorrentes

Conceitos



Recurrent Neural Network



As redes recorrentes possuem duas fontes de informação, o presente (novos dados) e o passado recente (estados internos).

Podemos assumir que as redes recorrentes possuem memória, compartilhando pesos ao longo do tempo.

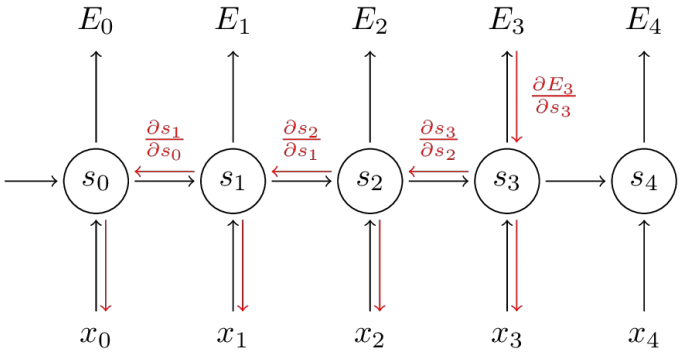
$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

O estado oculto é função da entrada de mesmo tempo modificada pelos pesos W adicionada ao estado oculto do passo de tempo anterior modificada pela matriz de transição U). Isso aplicado a uma função de transferência.

Backpropagation Through Time

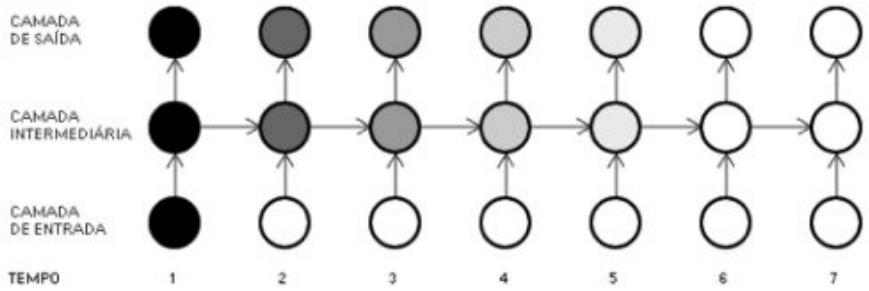
A atualização de pesos durante o treinamento em redes feedfowards é realizado pela retropropagação do erro das camadas finais para as camadas anteriores, por meio de uma “contribuição” dada pelas derivadas parciais do erro em função do seus parâmetros. (gradiente descendente)

No caso de redes recorrentes, a adição de mais funções aninhadas por causa da presença do processamento temporal, estende a regra da cadeia das derivadas parciais para uma extensão maior da série. Considera-se o erro total como a soma dos erros em cada etapa de tempo. Praticamente a mesma mecânica do backpropagation tradicional, quando se considera um formato da RNN “desenrolada”.



Dissipação do Gradiente

Por causa de longas cadeias de desdobramento do gradiente, causadas por longas sequências de processamento, o valor da “contribuição” do gradiente de dissipa durante os ajustes. Em outras palavras, as relações temporais de maior distância acabam não sendo assimiladas pelo modelo. Uma espécie de perda de memória de **longo prazo**.



RNN's e o problema da dissipação do gradiente (vanishing gradient problem)

O problema da dissipação do gradiente em redes profundas é minimizado com a utilização de funções Relu. Para as redes recorrentes foram projetadas as arquiteturas LSTM (1997) e as GRU (2014).

NLP Sequences

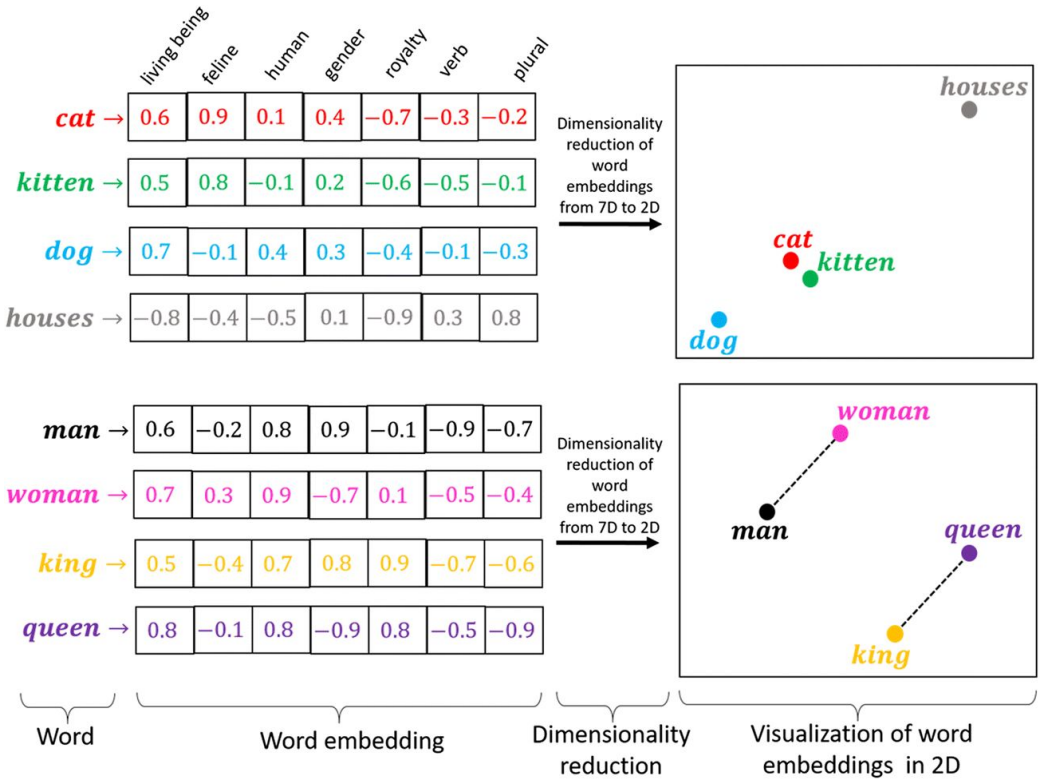
Vetorização de Textos

Representação numérica de textos para processamento matemático.

A construção vetorial pode representar aspectos:

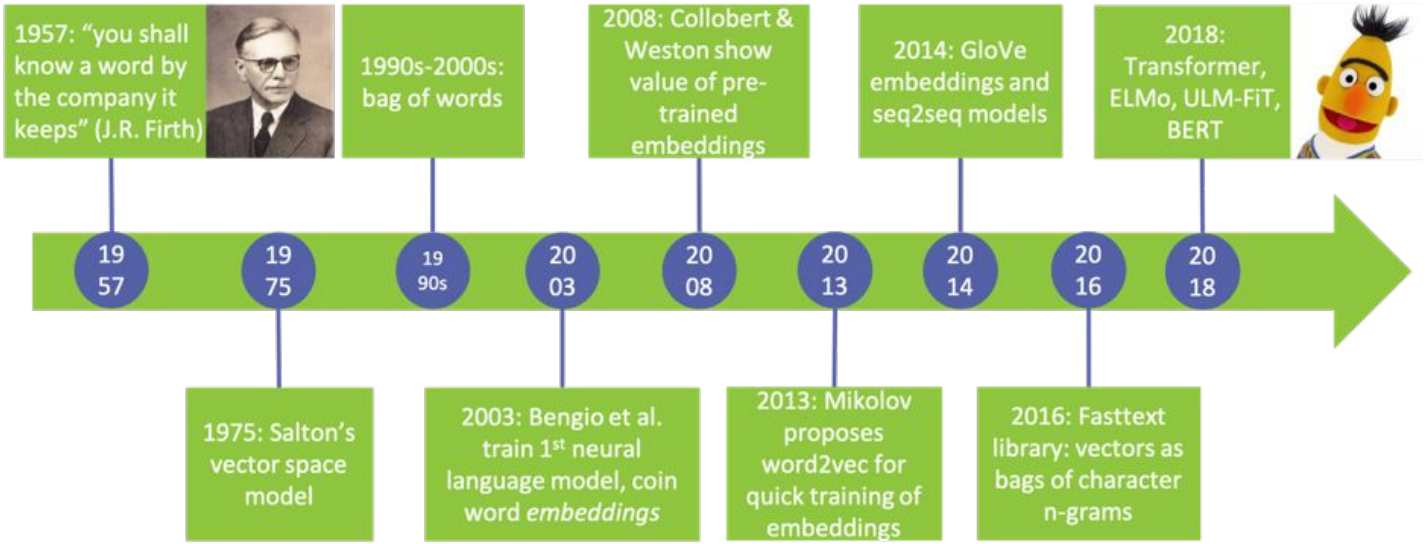
- Ortográficos
- Sintáticos
- Semânticos

Com a extração de features e construção de um espaço semântico latente.

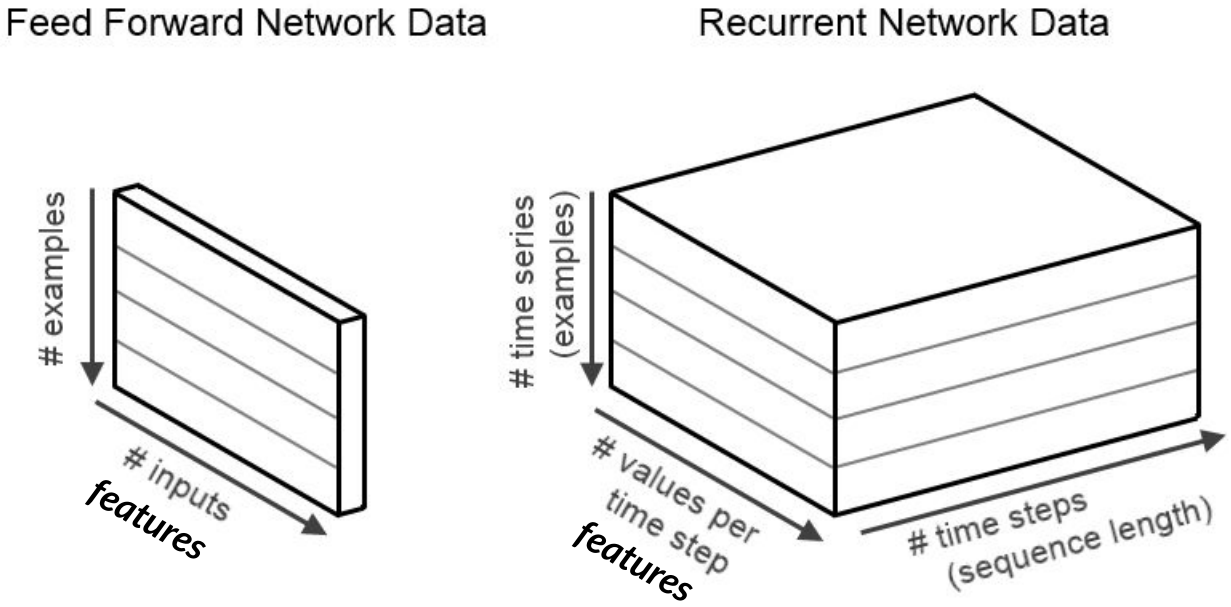


Text Embedding

Criação de representações vetoriais para textos (caracteres, palavras, frases ou parágrafos).



Formatação dos dados de entrada



Keras:

A 3D tensor with shape

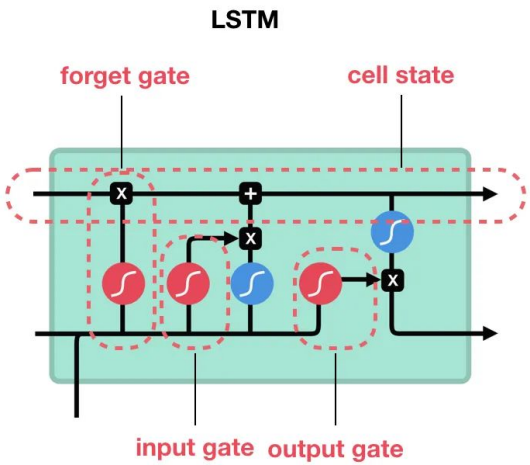
[batch, timesteps, feature].

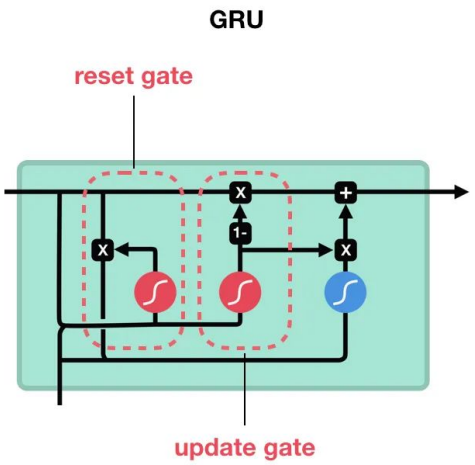
Gated Recurrent Unit (GRU)

Redes GRU

Consiste de uma variação da LSTM no entanto de forma mais simplificada.

Utiliza o estado oculto como célula de memória e possui apenas dois portões: reset e update capazes de manipular o estado oculto em função das memória de longo e curto prazo.





Redes GRU

Reset Gate:

- Porta definida pelo estado oculto da etapa anterior e os dados da entrada atual que define as informações a serem mantidas ou apagadas que devem ser combinadas para propor um novo estado oculto.

Update Gate:

- Porta também definida da relação do estado oculto da etapa anterior e os dados da entrada atual que define o quanto das informações passadas no estado oculto anterior precisa se retido para o futuro.

Backpropagation

- Um componente aditivo dos updates gates ajuda a manter parte do estado oculto em sua atualização, permitindo que erros do gradiente sejam propagados sem desaparecer ou explodir rapidamente.

Long Short-Term Memory (LSTM)

Redes LSTM

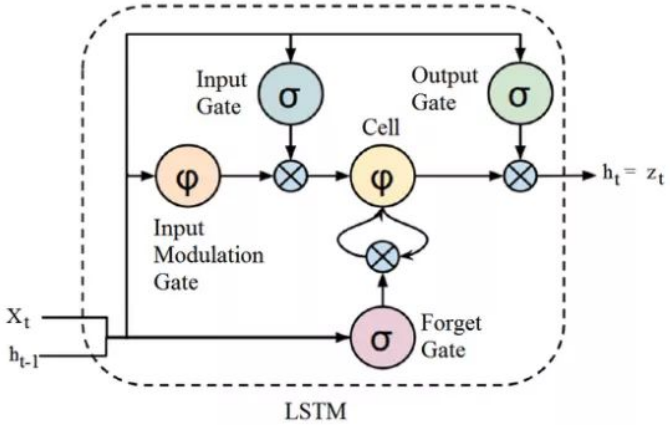
Utiliza blocos de memória conectados em camadas. Onde cada bloco possui “gates” de gerenciamento de estados que funcionam de forma análoga aos chips de memória: operações de escrita, leitura e de limpeza (reset) dos circuitos. A informação de longo prazo é retida pela célula, a informação de curto prazo pelo estado oculto e suas manipulações realizadas pelos gates.

Forget Gate: condicionalmente decide que informação deve ser descartada da unidade.

Input Gate: condicionalmente decide que valores do input que devem ter o estado de memória atualizado.

Output Gate: condicionalmente decide o que gerar como output com base no input e na memória da unidade.

Este bloco permite o armazenamento da informação e suas interações por longos períodos de tempo, pois não se degrada.



LSTM Stateful vs Stateless

Stateless:

- O estado oculto é mantido ao longo da sequência apresentada ao modelo neural e redefinida na mudança de batch ou amostras.
- Utilizado para casos onde cada batch é independente (ex: parágrafos)

Stateful:

- O estado oculto é mantido durante todo o batch / epóca de treinamento.
- Utilizado para casos onde os batchs possuem inter relações (ex: séries temporais)

```
# Cria o modelo LSTM
batch_size = 1
model = Sequential()
model.add(LSTM(4, batch_input_shape = (batch_size, look_back, 1), stateful = True))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
for i in range(200):
    model.fit(trainX, trainY, epochs = 1, batch_size = batch_size, verbose = 2, shuffle = False)
    model.reset_states()
```

Stacked LSTMs

Podemos criar camadas profundas de LSTMs da mesma maneira que as demais redes. Com a restrição de que a camada anterior a cada LSTM Layer retorne uma sequência (evolução dos estados internos).

```
# Cria o modelo LSTM
batch_size = 1
model = Sequential()
model.add(LSTM(4, batch_input_shape = (batch_size, look_back, 1), stateful = True, return_sequences = True))
model.add(LSTM(4, batch_input_shape = (batch_size, look_back, 1), stateful = True))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
for i in range(200):
    model.fit(trainX, trainY, epochs = 1, batch_size = batch_size, verbose = 2, shuffle = False)
    model.reset_states()
```

LSTMs e CNNs

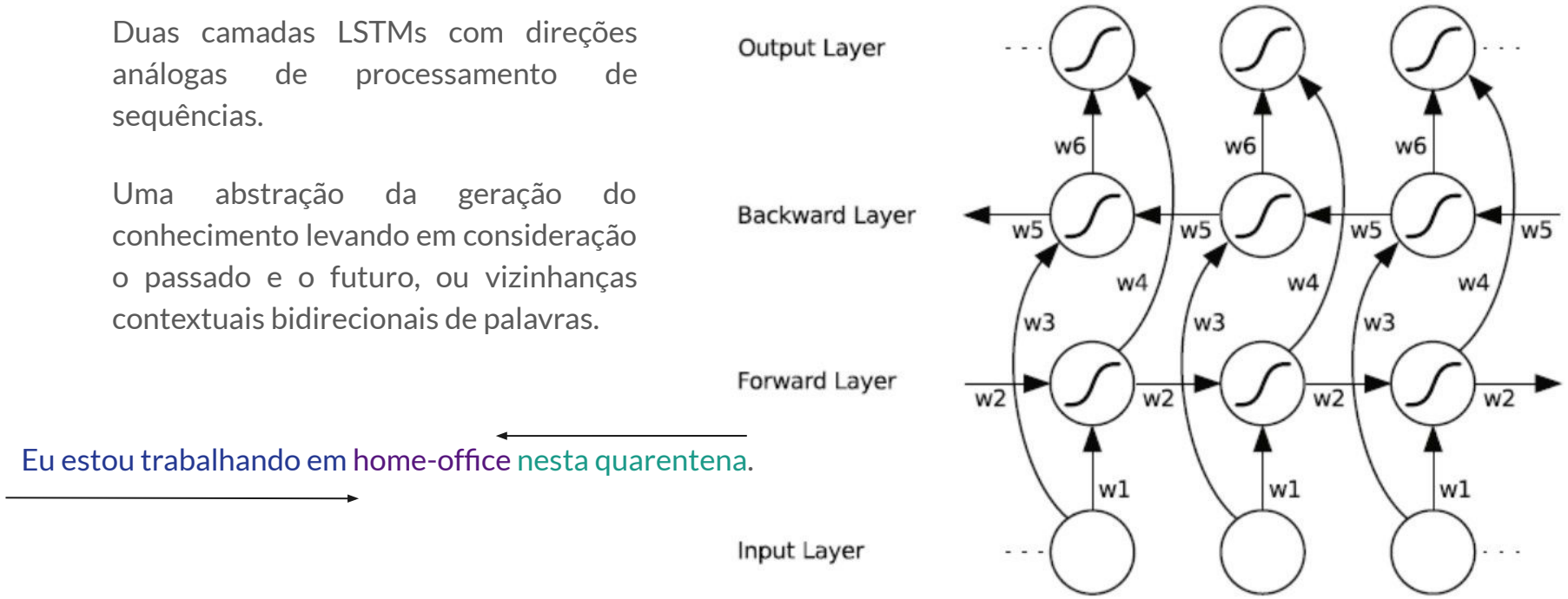
É possível integrar a extração de características de forma mesclada entre redes convolutivas e recorrentes.

```
# Cria o modelo
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length = max_review_length))
model.add(Convolution1D(filters = 32, kernel_size = 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = 2))
model.add(LSTM(100))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
print(model.summary())
model.fit(X_train, y_train, epochs = 3, batch_size = 64)
```

BILSTMs

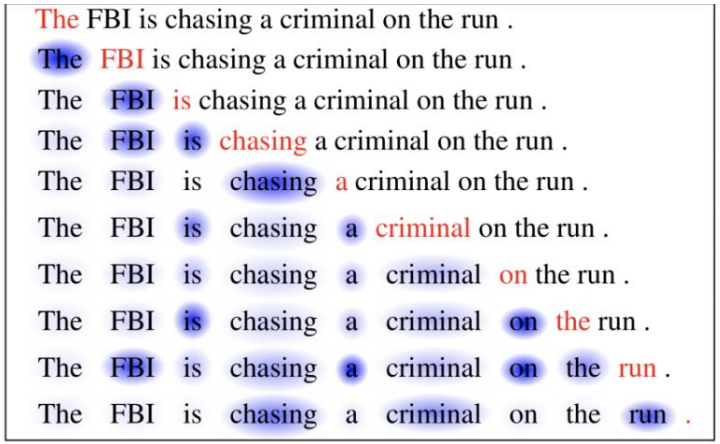
Duas camadas LSTMs com direções análogas de processamento de sequências.

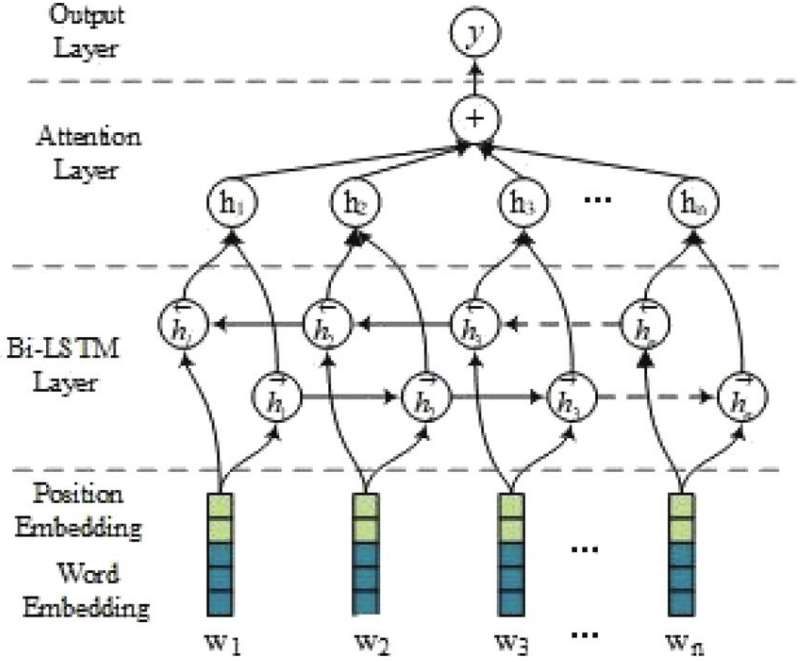
Uma abstração da geração do conhecimento levando em consideração o passado e o futuro, ou vizinhanças contextuais bidirecionais de palavras.



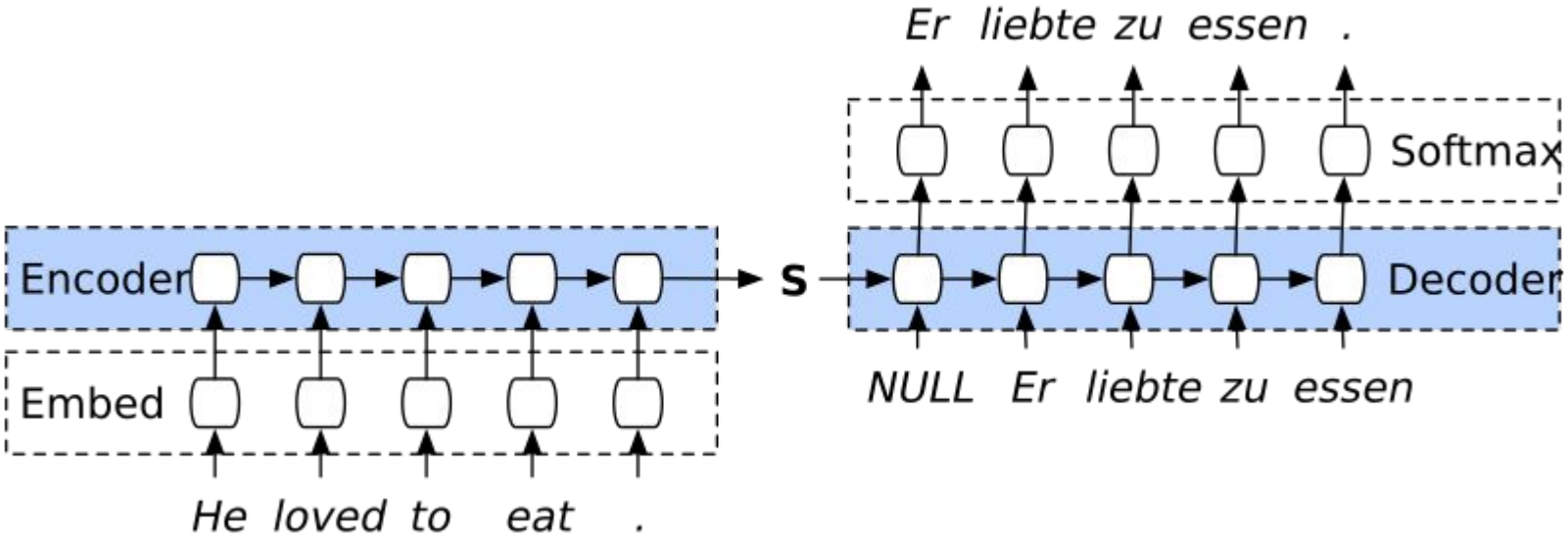
Mecanismos de Atenção

De forma bem simplificada, os mecanismos de atenção correspondem a camadas de processamento dos valores ocultos e intermediários de processamentos que buscam "*dar atenção*" aos momentos de maior relevância durante o consumo da sequência.



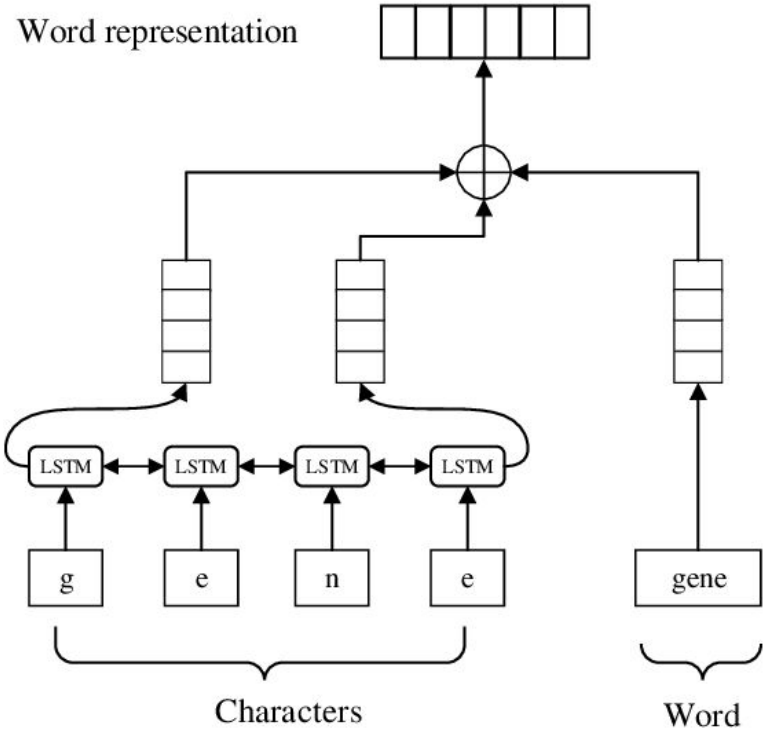


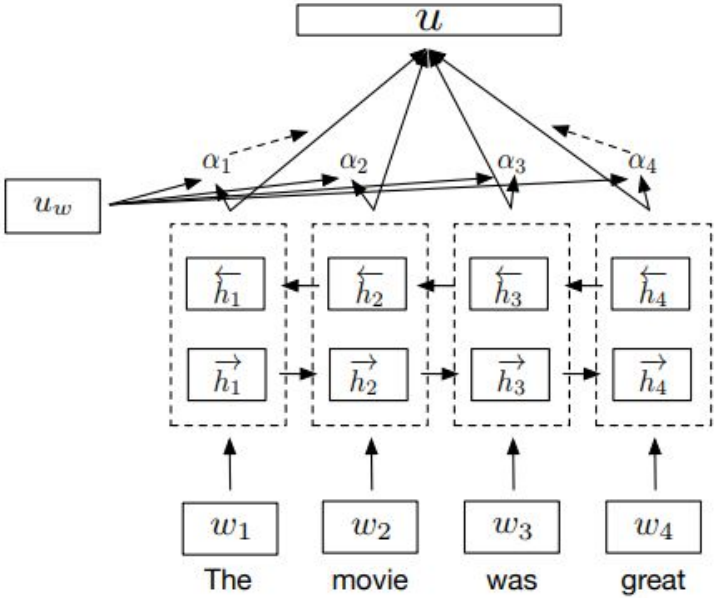
Encoder-Decoder Sequence Model



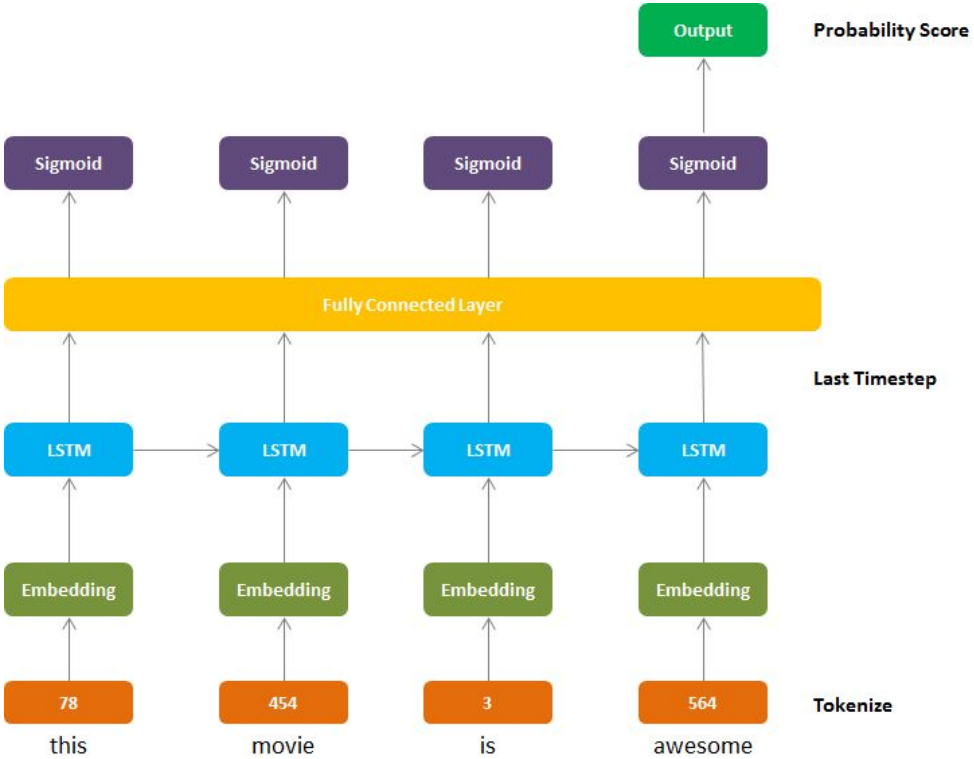
Estruturas para NLP

Char, Word and Sentence Embedding





Seq2Class



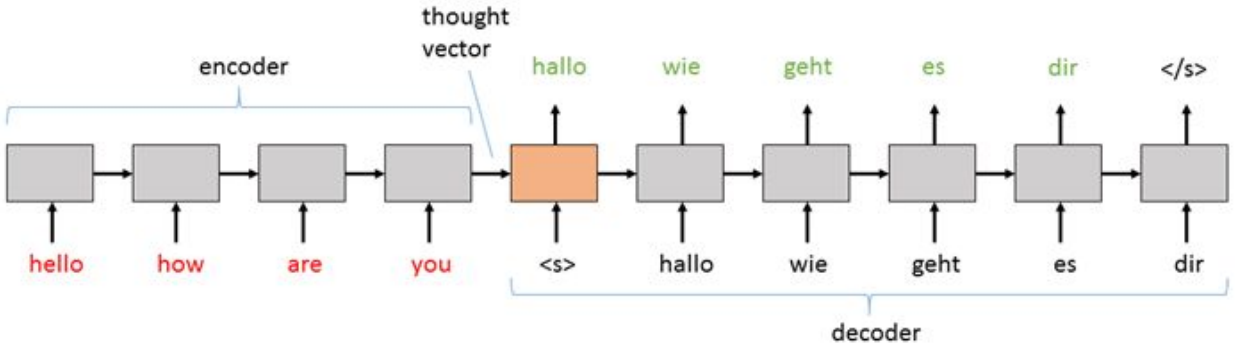
Estrutura

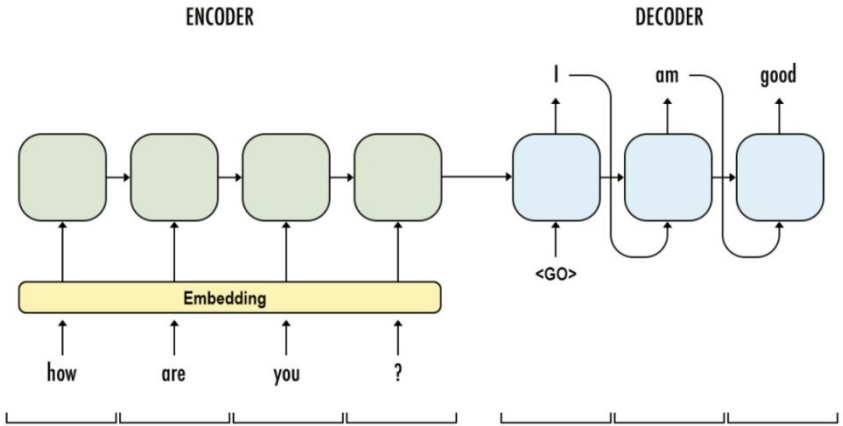
- Input: Sequência
- Output: Classificação

Exemplos:

- Análise de Sentimentos
- SPAM
- Named Entity Recognition

Seq2Seq





Estrutura

- Input: Sequência
- Output: Sequência

Exemplos:

- Tradução de Máquina
- QnA

Conclusões

Conclusões

- Devido a natureza do processamento sequencial do texto, as redes recorrentes são as topologias mais adequadas para o modelamento de problemas de Processamento de Linguagem Natural.
- As redes LSTM e BILSTM são soluções robustas para o processamento de sequências de informações sem perda do gradiente.
- Os mecanismos de atenção são complementações para facilitar a valorização de instantes temporais ao longo do processamento do modelo.
- As arquiteturas de redes recorrentes são as formas mais comuns de tratamento de problemas de sequência seja para embedding de informação, classificação de sequências ou geração de novas sequências.
- Precisamos sempre estar estudando para não perdermos o senso crítico sobre nossos modelos! =)

Agradecimentos

Oportunidades

vagas@omnilogic.ai

omnilogic.ai

omnilogic-ai

@omnilogic

@omnilogic_ai

@omnilogic.intelligence

@omnilogic_ai

Referências Bibliográficas

- Curso Deep Learning II - Data Science Academy
- https://www.asimovinstitute.org/neural-network-zoo/
- MacCartney B (2014) Understanding Natural Language Understanding, ACM SIGAI Bay Area Chapter Inaugural Meeting.
- Deep Learning Book (http://deeplearningbook.com.br)

Obrigado!

Perguntas e Discussões

Contatos:

Daniel Dominguete

daniel.dominguete@gmail.com

https://www.linkedin.com/in/dominguete/

Cognas

www.cognas.ai

Ringabell

www.ringabell.com.br