Transformers

Marcos Lopes

Departamento de Linguística – USP

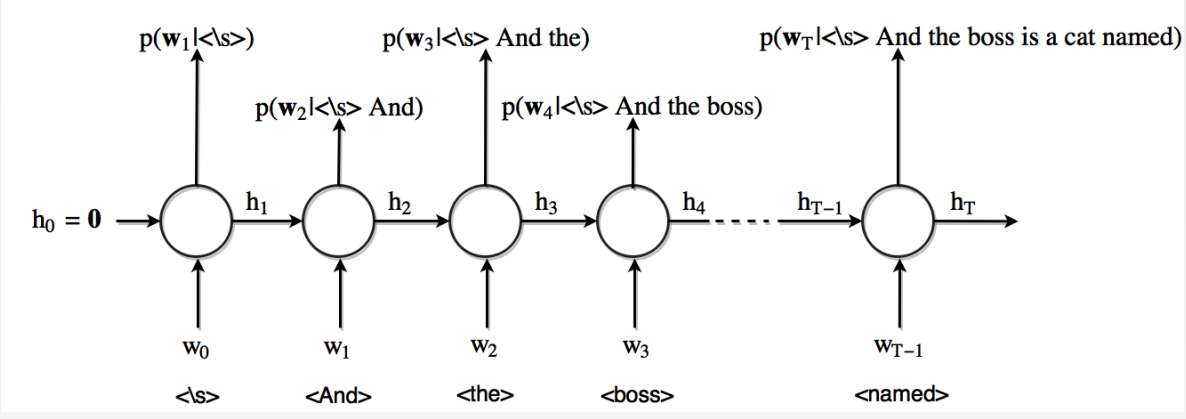
Limits das RNR

seq2seq

Transformers

BERT

Limits das RNR



Fonte: `https://blog.paperspace.com/recurrent-neural-networks-part-1-2/`

As redes neurais recorrentes (RNR) foram o paradigma dominante para todas as tarefas de PLN envolvendo sequências desde o seu surgimento até por volta de 2018.

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:
  - As representações de longa distância sofrem de um limite “prático” de propagação lateral (na camada recorrente): a longa distância não pode ser tão longa assim! Uma janela de umas poucas palavras representa esse limite prático, mesmo nas LSTMs.

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:
  - As representações de longa distância sofrem de um limite "prático" de propagação lateral (na camada recorrente): a longa distância não pode ser tão longa assim! Uma janela de umas poucas palavras representa esse limite prático, mesmo nas LSTMs.
  - As dependências bem distantes têm tendencialmente muito menos peso do que as próximas. Isso porque, além da dispersão do gradiente (que as "portas" da LSTM e da GRU tentam superar), a probabilidade linear da cadeia (que aparece nas sequências com janela $k = 1$ ou $k = 2$) é muito influente no cálculo dos pesos.

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:
  - As representações de longa distância sofrem de um limite "prático" de propagação lateral (na camada recorrente): a longa distância não pode ser tão longa assim! Uma janela de umas poucas palavras representa esse limite prático, mesmo nas LSTMs.
  - As dependências bem distantes têm tendencialmente muito menos peso do que as próximas. Isso porque, além da dispersão do gradiente (que as "portas" da LSTM e da GRU tentam superar), a probabilidade linear da cadeia (que aparece nas sequências com janela $k = 1$ ou $k = 2$) é muito influente no cálculo dos pesos.

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:
  - As representações de longa distância sofrem de um limite "prático" de propagação lateral (na camada recorrente): a longa distância não pode ser tão longa assim! Uma janela de umas poucas palavras representa esse limite prático, mesmo nas LSTMs.
  - As dependências bem distantes têm tendencialmente muito menos peso do que as próximas. Isso porque, além da dispersão do gradiente (que as "portas" da LSTM e da GRU tentam superar), a probabilidade linear da cadeia (que aparece nas sequências com janela $k = 1$ ou $k = 2$) é muito influente no cálculo dos pesos.

(1) Aquele menino telefonou. (curta distância; fácil)

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:
  - As representações de longa distância sofrem de um limite "prático" de propagação lateral (na camada recorrente): a longa distância não pode ser tão longa assim! Uma janela de umas poucas palavras representa esse limite prático, mesmo nas LSTMs.
  - As dependências bem distantes têm tendencialmente muito menos peso do que as próximas. Isso porque, além da dispersão do gradiente (que as "portas" da LSTM e da GRU tentam superar), a probabilidade linear da cadeia (que aparece nas sequências com janela $k = 1$ ou $k = 2$) é muito influente no cálculo dos pesos.

(1) Aquele menino telefonou. (curta distância; fácil)

(2) Aquele menino que veio aqui ontem telefonou. (média distância; não tão fácil)

Limits das RNR (cont.)

- Esses modelos têm limitações. Algumas delas, como a representação de dependências de longa distância, foram bastante suavizadas com o surgimento das LSTM, BiLSTM e GRU, mas parte dos problemas permanece:
  - As representações de longa distância sofrem de um limite "prático" de propagação lateral (na camada recorrente): a longa distância não pode ser tão longa assim! Uma janela de umas poucas palavras representa esse limite prático, mesmo nas LSTMs.
  - As dependências bem distantes têm tendencialmente muito menos peso do que as próximas. Isso porque, além da dispersão do gradiente (que as "portas" da LSTM e da GRU tentam superar), a probabilidade linear da cadeia (que aparece nas sequências com janela $k = 1$ ou $k = 2$) é muito influente no cálculo dos pesos.

(1) Aquele menino telefonou. (curta distância; fácil)

(2) Aquele menino que veio aqui ontem telefonou. (média distância; não tão fácil)

(3) Aquele menino que veio aqui ontem com um monte de amigos telefonou. (distância média-longa; bem mais difícil)

Limits das RNR (cont.)

- Conforme aumenta o tamanho da sequência, aumenta também o tempo de processamento, por conta do loop interno na camada de recorrência.

Limits das RNR (cont.)

- Conforme aumenta o tamanho da sequência, aumenta também o tempo de processamento, por conta do loop interno na camada de recorrência.
- Em outras palavras, o processamento nas RNR não pode ser paralelizado, porque é preciso esperar o processamento serial de todos os *tempos* da sequência.

Limits das RNR (cont.)

- Conforme aumenta o tamanho da sequência, aumenta também o tempo de processamento, por conta do loop interno na camada de recorrência.
- Em outras palavras, o processamento nas RNR não pode ser paralelizado, porque é preciso esperar o processamento serial de todos os *tempos* da sequência.
- Como consequência, é impraticável treinar essas redes em conjuntos de dados muito grandes.

Limits das RNR (cont.)

- Conforme aumenta o tamanho da sequência, aumenta também o tempo de processamento, por conta do loop interno na camada de recorrência.
- Em outras palavras, o processamento nas RNR não pode ser paralelizado, porque é preciso esperar o processamento serial de todos os *tempos* da sequência.
- Como consequência, é impraticável treinar essas redes em conjuntos de dados muito grandes.
- O problema é ainda mais agravado quando se pensa que as GPUs e TPUs são especialmente boas para a paralelização de tarefas, o que as RNRs não podem aproveitar.

Modelos sequência-a-sequência

- Um modelo sequência-a-sequência *(sequence-to-sequence* ou *seq2seq)* tem como entrada uma sequência e, como saída, outra sequência.

Modelos sequência-a-sequência

- Um modelo sequência-a-sequência *(sequence-to-sequence* ou *seq2seq)* tem como entrada uma sequência e, como saída, outra sequência.
    - Compare com um classificador binário baseado em sequências, em que se tem uma sequência na entrada (como o texto de uma resenha) mas não na saída, onde se espera somente um número (0 ou 1).

Modelos sequência-a-sequência

- Um modelo sequência-a-sequência *(sequence-to-sequence* ou *seq2seq)* tem como entrada uma sequência e, como saída, outra sequência.
    - Compare com um classificador binário baseado em sequências, em que se tem uma sequência na entrada (como o texto de uma resenha) mas não na saída, onde se espera somente um número (0 ou 1).
- No PLN, as sequências podem ser constituídas de letras, pedaços de palavras, palavras ou sentenças inteiras.

Modelos sequência-a-sequência

- Um modelo sequência-a-sequência *(sequence-to-sequence* ou *seq2seq)* tem como entrada uma sequência e, como saída, outra sequência.
    - Compare com um classificador binário baseado em sequências, em que se tem uma sequência na entrada (como o texto de uma resenha) mas não na saída, onde se espera somente um número (0 ou 1).
- No PLN, as sequências podem ser constituídas de letras, pedaços de palavras, palavras ou sentenças inteiras.
    - Os tokens dessas sequências são, respetivamente, letras, pedaços de palavras etc.

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)
- São, basicamente, duas RNR acopladas: tanto o codificador quanto o decodificador são RNRs.

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)
- São, basicamente, duas RNR acopladas: tanto o codificador quanto o decodificador são RNRs.
- Consideremos que nossos tokens sejam palavras. Nesse caso, a sequência de entrada são os word embeddings de cada palavra da sequência.

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)
- São, basicamente, duas RNR acopladas: tanto o codificador quanto o decodificador são RNRs.
- Consideremos que nossos tokens sejam palavras. Nesse caso, a sequência de entrada são os word embeddings de cada palavra da sequência.
- O codificador compila a informação da entrada sequenciada num vetor, que recebe o nome de “contexto”.

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)
- São, basicamente, duas RNR acopladas: tanto o codificador quanto o decodificador são RNRs.
- Consideremos que nossos tokens sejam palavras. Nesse caso, a sequência de entrada são os word embeddings de cada palavra da sequência.
- O codificador compila a informação da entrada sequenciada num vetor, que recebe o nome de “contexto”.
- O contexto é um vetor de números reais *(floats)*. O tamanho do contexto corresponde ao tamanho da camada escondida do codificador.

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)
- São, basicamente, duas RNR acopladas: tanto o codificador quanto o decodificador são RNRs.
- Consideremos que nossos tokens sejam palavras. Nesse caso, a sequência de entrada são os word embeddings de cada palavra da sequência.
- O codificador compila a informação da entrada sequenciada num vetor, que recebe o nome de “contexto”.
- O contexto é um vetor de números reais *(floats)*. O tamanho do contexto corresponde ao tamanho da camada escondida do codificador.
- O contexto é passado ao decodificador, que produz a sequência de saída token por token.

Arquitetura Encoder-Decoder

- Os modelos sequência-a-sequência geralmente usam arquiteturas *encoder – decoder* (codificador – decodificador)
- São, basicamente, duas RNR acopladas: tanto o codificador quanto o decodificador são RNRs.
- Consideremos que nossos tokens sejam palavras. Nesse caso, a sequência de entrada são os word embeddings de cada palavra da sequência.
- O codificador compila a informação da entrada sequenciada num vetor, que recebe o nome de "contexto".
- O contexto é um vetor de números reais *(floats)*. O tamanho do contexto corresponde ao tamanho da camada escondida do codificador.
- O contexto é passado ao decodificador, que produz a sequência de saída token por token.
- Observe que o contexto que vai para o decodificador corresponde à informação somente da última camada escondida do codificador (h_T).

Modelos Transformers

- Propostos por Vaswany et al. (2017).

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).
- A atenção incorporada ao modelo, que permite representar muito mais informação contextual, é aquilo que faz a grande diferença por relação aos modelos Encoder – Decoder.

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).
- A atenção incorporada ao modelo, que permite representar muito mais informação contextual, é aquilo que faz a grande diferença por relação aos modelos Encoder – Decoder.
- Seis codificadores e seis decodificadores são colocados na arquitetura.

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).
- A atenção incorporada ao modelo, que permite representar muito mais informação contextual, é aquilo que faz a grande diferença por relação aos modelos Encoder – Decoder.
- Seis codificadores e seis decodificadores são colocados na arquitetura.
- Os codificadores são todos iguais. Todos são compostos por uma camada de autoatenção (detalhes no próximo slide) que alimenta uma rede Feedforward.

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).
- A atenção incorporada ao modelo, que permite representar muito mais informação contextual, é aquilo que faz a grande diferença por relação aos modelos Encoder – Decoder.
- Seis codificadores e seis decodificadores são colocados na arquitetura.
- Os codificadores são todos iguais. Todos são compostos por uma camada de autoatenção (detalhes no próximo slide) que alimenta uma rede Feedforward.
- Nos decodificadores, além dessas duas camadas, uma camada intermediária de atenção (com codificação-decodificação) aparece entre elas.

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).
- A atenção incorporada ao modelo, que permite representar muito mais informação contextual, é aquilo que faz a grande diferença por relação aos modelos Encoder – Decoder.
- Seis codificadores e seis decodificadores são colocados na arquitetura.
- Os codificadores são todos iguais. Todos são compostos por uma camada de autoatenção (detalhes no próximo slide) que alimenta uma rede Feedforward.
- Nos decodificadores, além dessas duas camadas, uma camada intermediária de atenção (com codificação-decodificação) aparece entre elas.
- O primeiro dos seis codificadores recebe como entrada os word embeddings (512 dimensões) das palavras.

Modelos Transformers

- Propostos por Vaswany et al. (2017).
- Nessa proposta inicial, tratava-se de uma arquitetura Encoder-Decoder com *atenção* (ver a seguir).
- A atenção incorporada ao modelo, que permite representar muito mais informação contextual, é aquilo que faz a grande diferença por relação aos modelos Encoder – Decoder.
- Seis codificadores e seis decodificadores são colocados na arquitetura.
- Os codificadores são todos iguais. Todos são compostos por uma camada de autoatenção (detalhes no próximo slide) que alimenta uma rede Feedforward.
- Nos decodificadores, além dessas duas camadas, uma camada intermediária de atenção (com codificação-decodificação) aparece entre elas.
- O primeiro dos seis codificadores recebe como entrada os word embeddings (512 dimensões) das palavras.
- As palavras (isto é, seus embeddings) das sequências são processadas em paralelo nos codificadores.

Atenção

- A arquitetura Encoder-Decoder pode ser aprimorada (quanto às limitações já mencionadas de todas as RNRs) com a introdução de *atenção*.

Atenção

- A arquitetura Encoder-Decoder pode ser aprimorada (quanto às limitações já mencionadas de todas as RNRs) com a introdução de *atenção*.
- A atenção é um método de se estimar que partes do contexto devem receber mais peso no processamento.

Atenção

- A arquitetura Encoder-Decoder pode ser aprimorada (quanto às limitações já mencionadas de todas as RNRs) com a introdução de *atenção*.
- A atenção é um método de se estimar que partes do contexto devem receber mais peso no processamento.
- Cada palavra de uma sequência é comparada a todas as outras na mesma sequência, incluindo ela mesma, refazendo os pesos dos word embeddings com a informação contextual.

Atenção

- A arquitetura Encoder-Decoder pode ser aprimorada (quanto às limitações já mencionadas de todas as RNRs) com a introdução de *atenção*.
- A atenção é um método de se estimar que partes do contexto devem receber mais peso no processamento.
- Cada palavra de uma sequência é comparada a todas as outras na mesma sequência, incluindo ela mesma, refazendo os pesos dos word embeddings com a informação contextual.
- A isso se chama *Contextualized Word Embeddings*.

Atenção

- A arquitetura Encoder-Decoder pode ser aprimorada (quanto às limitações já mencionadas de todas as RNRs) com a introdução de *atenção*.
- A atenção é um método de se estimar que partes do contexto devem receber mais peso no processamento.
- Cada palavra de uma sequência é comparada a todas as outras na mesma sequência, incluindo ela mesma, refazendo os pesos dos word embeddings com a informação contextual.
- A isso se chama *Contextualized Word Embeddings*.
- Essa é uma forma de resolver os problemas relacionados à polissemia, à sinonímia, à homonímia, enfim, a todas as questões ligadas ao significado do léxico.

Atenção

- A arquitetura Encoder-Decoder pode ser aprimorada (quanto às limitações já mencionadas de todas as RNRs) com a introdução de *atenção*.
- A atenção é um método de se estimar que partes do contexto devem receber mais peso no processamento.
- Cada palavra de uma sequência é comparada a todas as outras na mesma sequência, incluindo ela mesma, refazendo os pesos dos word embeddings com a informação contextual.
- A isso se chama *Contextualized Word Embeddings*.
- Essa é uma forma de resolver os problemas relacionados à polissemia, à sinonímia, à homonímia, enfim, a todas as questões ligadas ao significado do léxico.
- Dessa forma, se você tem a palavra “banco” numa sequência e não sabe qual o seu contexto, não dá para saber se o texto fala de economia (banco como instituição financeira) ou de lazer (banco da praça). (ver WordNet)

Atenção (cont.)

- Vimos que o contexto é incorporado em cada palavra que participa dele. Mas como isso é feito?

Atenção (cont.)

- Vimos que o contexto é incorporado em cada palavra que participa dele. Mas como isso é feito?
- Nos modelos que usam atenção:

Atenção (cont.)

- Vimos que o contexto é incorporado em cada palavra que participa dele. Mas como isso é feito?
- Nos modelos que usam atenção:
  - O *codificador* passa muito mais informações ao decodificador: não só o estado da última camada escondida (h_T), mas todos ($h_{0..T}$).

Atenção (cont.)

- Vimos que o contexto é incorporado em cada palavra que participa dele. Mas como isso é feito?
- Nos modelos que usam atenção:
  - O *codificador* passa muito mais informações ao decodificador: não só o estado da última camada escondida (h_T), mas todos ($h_{0..T}$).
  - O *decodificador* atribui valores adicionais aos estados recebidos, aplica uma função Softmax a cada um deles e multiplica os vetores pelo resultado da Softmax, gerando um conjunto de vetores ponderados. Por fim, esses vetores são somados para produzir o contexto de saída.

Autoatenção

Os word embeddings de cada palavra são usados para criar três vetores (de 64 dimensões) através da multiplicação dos embeddings por matrizes obtidas no treinamento: Vetor de consulta *(query)*; Vetor chave *key* e Vetor de valor *value*.

Autoatenção

Os word embeddings de cada palavra são usados para criar três vetores (de 64 dimensões) através da multiplicação dos embeddings por matrizes obtidas no treinamento: Vetor de consulta *(query)*; Vetor chave *key* e Vetor de valor *value*.

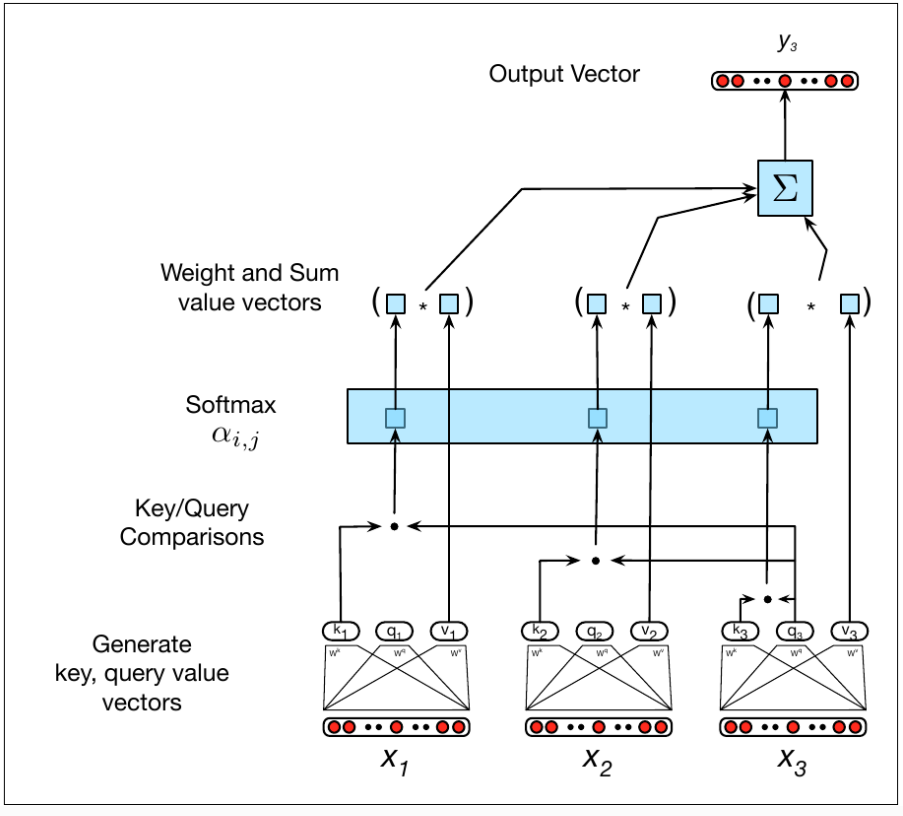


Figura 1: Cálculo do terceiro elemento de uma sequência usando auto-atenção. Fonte: Jurafsky & Martin (2020).

(Transformer Block)

- A camada de atenção é um dos componentes do Bloco Transformer *(Transformer Block)*, como ilustrado a seguir.

(Transformer Block)

- A camada de atenção é um dos componentes do Bloco Transformer *(Transformer Block)*, como ilustrado a seguir.
- *n* blocos podem ser empilhados em série, exatamente como camadas de redes recorrentes.

(Transformer Block)

- A camada de atenção é um dos componentes do Bloco Transformer *(Transformer Block)*, como ilustrado a seguir.
- *n* blocos podem ser empilhados em série, exatamente como camadas de redes recorrentes.

(Transformer Block)

- A camada de atenção é um dos componentes do Bloco Transformer *(Transformer Block)*, como ilustrado a seguir.
- *n* blocos podem ser empilhados em série, exatamente como camadas de redes recorrentes.

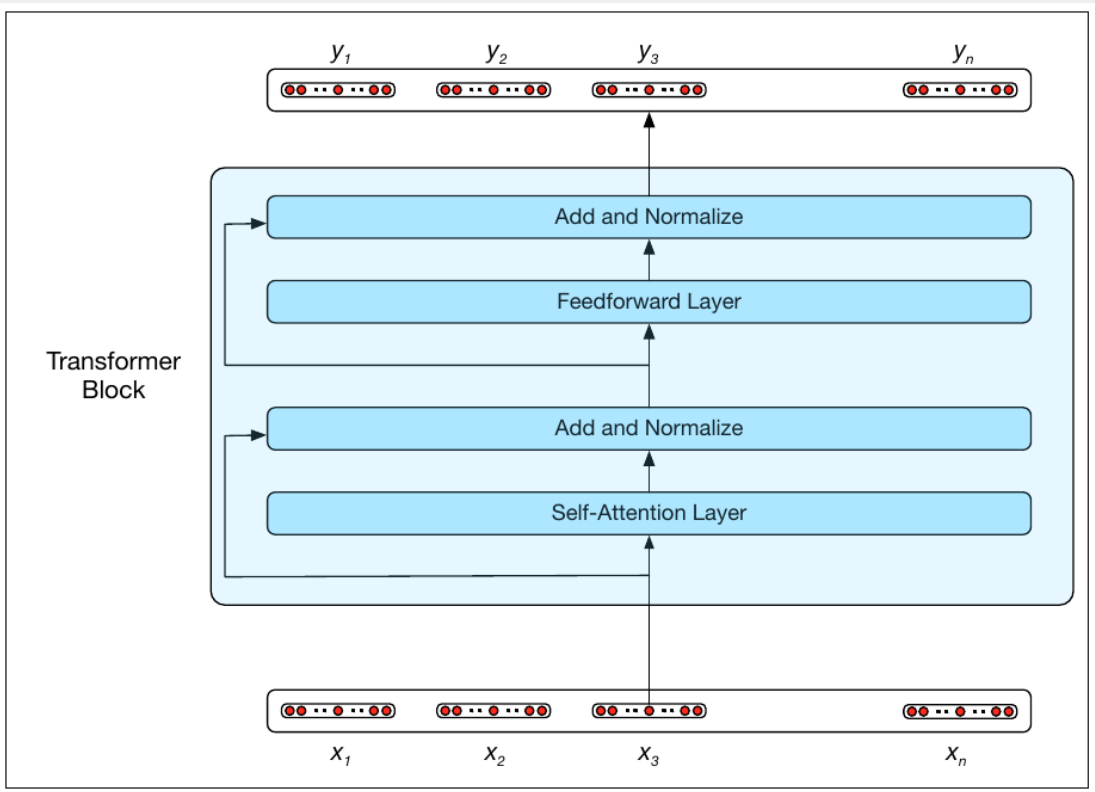


Figura 2: Arquitetura interna de um Bloco Transformer. Fonte: Jurafsky & Martin

Atenção multi-nucleada (*Multi-Headed Attention*)

- Conforme a necessidade de representar contextos cada vez mais abrangentes (uma página inteira de texto ou mais, por exemplo) vai aumentando, a auto-atenção pode não ser suficiente.

Atenção multi-nucleada (*Multi-Headed Attention*)

- Conforme a necessidade de representar contextos cada vez mais abrangentes (uma página inteira de texto ou mais, por exemplo) vai aumentando, a auto-atenção pode não ser suficiente.
- Através de múltiplos núcleos de atenção, entretanto, o modelo torna-se capaz de focar em informações de diversas partes do texto.

Atenção multi-nucleada (*Multi-Headed Attention*)

- Conforme a necessidade de representar contextos cada vez mais abrangentes (uma página inteira de texto ou mais, por exemplo) vai aumentando, a auto-atenção pode não ser suficiente.
- Através de múltiplos núcleos de atenção, entretanto, o modelo torna-se capaz de focar em informações de diversas partes do texto.
- Os núcleos de atenção funcionam em paralelo e não compartilham os pesos durante o processamento.

Atenção multi-nucleada (*Multi-Headed Attention*)

- Conforme a necessidade de representar contextos cada vez mais abrangentes (uma página inteira de texto ou mais, por exemplo) vai aumentando, a auto-atenção pode não ser suficiente.
- Através de múltiplos núcleos de atenção, entretanto, o modelo torna-se capaz de focar em informações de diversas partes do texto.
- Os núcleos de atenção funcionam em paralelo e não compartilham os pesos durante o processamento.
- Ao final, os vetores de saída de todos os núcleos são concatenados num único vetor.

Limits das RNR

seq2seq

Transformers

BERT

O que é BERT?

- BERT: Bidirectional Encoder Representations from Transformers (2018)

O que é BERT?

- BERT: Bidirectional Encoder Representations from Transformers (2018)
- Modelos pré-treinados em grandes quantidades de dados de língua natural (Wikipedia e livros) usando uma combinação entre modelagem mascarada de linguagem *masked language modelling* e previsão de próxima sentença.

O que é BERT?

- BERT: Bidirectional Encoder Representations from Transformers (2018)
- Modelos pré-treinados em grandes quantidades de dados de língua natural (Wikipedia e livros) usando uma combinação entre modelagem mascarada de linguagem *masked language modelling* e previsão de próxima sentença.
- Inicialmente, foram gerados dois modelos pelos autores:

O que é BERT?

- BERT: Bidirectional Encoder Representations from Transformers (2018)
- Modelos pré-treinados em grandes quantidades de dados de língua natural (Wikipedia e livros) usando uma combinação entre modelagem mascarada de linguagem *masked language modelling* e previsão de próxima sentença.
- Inicialmente, foram gerados dois modelos pelos autores:
    - BERT large (pré-treinado com 345 M de parâmetros)

O que é BERT?

- BERT: Bidirectional Encoder Representations from Transformers (2018)
- Modelos pré-treinados em grandes quantidades de dados de língua natural (Wikipedia e livros) usando uma combinação entre modelagem mascarada de linguagem *masked language modelling* e previsão de próxima sentença.
- Inicialmente, foram gerados dois modelos pelos autores:
  - BERT large (pré-treinado com 345 M de parâmetros)
  - BERT base (110 M de parâmetros))

O que é BERT?

- BERT: Bidirectional Encoder Representations from Transformers (2018)
- Modelos pré-treinados em grandes quantidades de dados de língua natural (Wikipedia e livros) usando uma combinação entre modelagem mascarada de linguagem *masked language modelling* e previsão de próxima sentença.
- Inicialmente, foram gerados dois modelos pelos autores:
  - BERT large (pré-treinado com 345 M de parâmetros)
  - BERT base (110 M de parâmetros))
- A arquitetura usa uma pilha de 12 (modelo básico) ou 24 (grande) codificadores transformers, com camadas escondidas de 768 (básico) e 1024 (grande) unidades, e 12 ou 16 núcleos de atenção *(attention heads)*.

O que é BERT? (cont.)

- Os tokens, para o BERT, não são simples palavras e pontuação. São pedaços de palavras *(WordPieces)* gerados pelo tokenizador.

O que é BERT? (cont.)

- Os tokens, para o BERT, não são simples palavras e pontuação. São pedaços de palavras *(WordPieces)* gerados pelo tokenizador.
- São acrescentados tokens especiais aos embeddings de entrada: [CLS] no início das sequências a classificar, e [SEP] como delimitador final.

O que é BERT? (cont.)

- Os tokens, para o BERT, não são simples palavras e pontuação. São pedaços de palavras *(WordPieces)* gerados pelo tokenizador.
- São acrescentados tokens especiais aos embeddings de entrada: [CLS] no início das sequências a classificar, e [SEP] como delimitador final.
- Como saída, são gerados vetores do tamanho da camada escondida (768 ou 1024) para cada token de entrada.

O que é BERT? (cont.)

- Os tokens, para o BERT, não são simples palavras e pontuação. São pedaços de palavras *(WordPieces)* gerados pelo tokenizador.
- São acrescentados tokens especiais aos embeddings de entrada: [CLS] no início das sequências a classificar, e [SEP] como delimitador final.
- Como saída, são gerados vetores do tamanho da camada escondida (768 ou 1024) para cada token de entrada.
- Nas tarefas de classificação, geralmente se considera somente os embeddings do token [CLS] da última camada do modelo. Ele incorpora informações contextuais de todos os demais tokens da sequência, naturalmente.

O que é BERT? (cont.)

- Os tokens, para o BERT, não são simples palavras e pontuação. São pedaços de palavras *(WordPieces)* gerados pelo tokenizador.
- São acrescentados tokens especiais aos embeddings de entrada: [CLS] no início das sequências a classificar, e [SEP] como delimitador final.
- Como saída, são gerados vetores do tamanho da camada escondida (768 ou 1024) para cada token de entrada.
- Nas tarefas de classificação, geralmente se considera somente os embeddings do token [CLS] da última camada do modelo. Ele incorpora informações contextuais de todos os demais tokens da sequência, naturalmente.
- A seguir, esses embeddings podem ser usados por um modelo "clássico" de classificação, como a regressão logística ou qualquer outro.