UNIVERSIDADE FEDERAL DA GRANDE DOURADOS

Thiago Lourenço dos Santos

Sumarização Automática Abstrativa de Textos utilizando *Deep Learning*

DOURADOS – MS

2020

Thiago Lourenço dos Santos

Sumarização Automática Abstrativa de Textos utilizando *Deep Learning*

Trabalho de Conclusão de Curso de graduação apresentado para obtenção do título de Bacharel em Engenharia de Computação pela Faculdade de Ciências Exatas e Tecnologia da Universidade Federal da Grande Dourados.

Orientador: Prof.° Dr. Joinvile Batista Junior

DOURADOS – MS

2020

Dourados, 20 de agosto de 2020.

Dados Internacionais de Catalogação na Publicação (CIP).

S237s Santos, Thiago Lourenco Dos

Sumarização Automática Abstrativa de Textos utilizando Deep Learning [recurso eletrônico] / Thiago Lourenco Dos Santos. -- 2020.

Arquivo em formato pdf.

Orientador: Joinvile Batista Junior.

TCC (Graduação em Engenharia de Computação)-Universidade Federal da Grande Dourados, 2020.

Disponível no Repositório Institucional da UFGD em: https://portal.ufgd.edu.br/setor/biblioteca/repositorio

1. deep learning. 2. processamento de linguagem natural. 3. sumarização abstrativa. I. Batista Junior, Joinvile . II. Título.

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

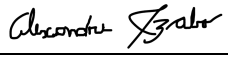
Thiago Lourenço dos Santos

Sumarização Automática Abstrativa de Textos utilizando *Deep Learning*

Trabalho de Conclusão de Curso aprovado como requisito para obtenção do título de Bacharel em Engenharia de Computação na Universidade Federal da Grande Dourados, pela comissão formada por:

Orientador Prof. Dr. Joinvile Batista Junior

FACET – UFGD



Prof. Dr. Alexandre Szabo

FACET – UFGD

Prof. Dr. Wellington Lima dos Santos

FACET – UFGD

Dourados, 20 de agosto de 2020.

Resumo

Deep learning (aprendizado profundo) tem sido utilizado com sucesso em várias publicações na área de Processamento de Linguagem Natural. Uma das aplicações é a sumarização de sentenças, que pode ser extrativa ou abstrativa. Neste trabalho será abordada a sumarização abstrativa, na qual o sumário é gerado a partir de abstrações e técnicas linguísticas, com a possibilidade de uso de palavras não encontradas no texto de entrada. O que não ocorre na sumarização extrativa, na qual a saída é um subconjunto da entrada. Para isso, foram selecionados cinco artigos, em que as implementações dos dois artigos mais recentes foram executados e testados. Os resultados foram comparados com os divulgados nos artigos. Além disso, foi feita uma análise comparativa de alguns dos sumários gerados pelos modelos treinados.

Palavras-Chave: *deep learning*, processamento de linguagem natural, sumarização abstrativa.

Abstract

Deep learning has been used successfully in several applications in Natural Language Processing area. One of the applications is summarization of sentences, which can be extractive or abstractive. In this work, abstract summarization is addressed, on which summary is generated from abstractions and linguistic techniques, with the possibility of using words not found in the input sentence, which does not occur in extractive summarization, on which the output is a subset of the input. In this work, five papers were selected, in which the two most recent articles implementations were executed and tested, with the results being compared with those published in the articles and with the comparative analysis of some of the summaries generated by the trained models.

Key-Words: Deep learning, Natural Language Processing, Abstractive Summarization.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

1. INTRODUÇÃO

Atualmente, a sociedade vivencia a era da informação, no qual o rápido acesso das tecnologias de comunicação transformou a maneira como as relações sociais e profissionais são desenvolvidas (Hilbert, 2012). Com o advento de novas tecnologias, o volume de informações disponíveis na *web* cresceu exponencialmente. Dessa maneira, é impossível conseguir acompanhar e resumir um fluxo tão grande de informações (TAS, 2007).

Por essa razão, estudos sobre formas de sintetizar essas informações têm angariado relevância em publicações científicas da área de Processamento de Linguagem Natural (PLN). Uma área de PLN que têm ganhado certo destaque é a sumarização automática de textos que, segundo Mani et al. (1999), é definida como o processo de refinar as informações mais importantes de uma ou mais fontes e produzir uma versão resumida para usuários ou tarefas específicas.

Assim sendo, sumarização automática de textos é uma tarefa, que pode ser dividida entre dois tipos de abordagem: extrativa e abstrativa. Na sumarização extrativa, o sumário é gerado extraindo as sentenças mais relevantes do texto fonte, sem modificações. Na sumarização abstrativa são utilizadas técnicas linguísticas para examinar e captar as informações essenciais do texto fonte, produzindo um sumário que generalize as sentenças de entrada (Moratanch, 2016), com a possibilidade de utilizar palavras não encontradas no texto fonte.

A abordagem abstrativa recebeu maior atenção após trabalhos na área de tradução de máquina neural (NMT, *Neural Machine Translation*) utilizando *deep learning* (aprendizado profundo), nos quais são utilizadas redes com múltiplas camadas de neurônios (profundidade), apresentarem resultados promissores, ditando a direção das publicações posteriores de sumarização abstrativa (Rush et al., 2015; Nallapati et al., 2016). O foco deste trabalho é a sumarização automática abstrativa utilizando *deep learning*.

No capítulo 2, são apresentadas as fundamentações teóricas do trabalho, com descrições dos modelos propostos nos artigos selecionados. No capítulo 3, é descrito o desenvolvimento realizado englobando: os critérios de seleção dos artigos, a escolha dos artigos prototipados, a execução de treinamentos dos modelos envolvidos e as

comparações com os resultados relatados nos artigos. No capítulo 4, são apresentadas as conclusões, dificuldades encontradas durante o desenvolvimento do trabalho, e indicações de trabalhos futuros.

1.2 OBJETIVOS DO TRABALHO

O objetivo geral deste trabalho é a reprodução de experimentos com aplicação de *deep learning* na área de sumarização abstrativa de textos.

Os objetivos específicos são:

- elucidar termos comuns do espaço do problema trabalhado;
- identificar artigos da área de sumarização abstrativa que utilizem *deep learning*;
- definir critérios de seleção que servirão de base para a escolha de cinco artigos para análise;
- executar o treinamento dos dois modelos mais recentes;
- comparar o resultado final com os dados apresentados nos artigos;
- analisar os resultados gerados.

1.3 JUSTIFICATIVA

A sumarização automática tem grande relevância na extração das informações relevantes de um texto, pois permite que seja possível selecionar o que vai ser lido e transformado em conhecimento. E o *deep learning*, um dos ramos da área de aprendizado de máquina, tem agregado valor em várias áreas de conhecimento como, por exemplo: visão computacional e PLN (Processamento de Linguagem Natural). E dentro desse contexto, temos a sumarização abstrativa. A escolha da abordagem abstrativa para este trabalho é justificada porque ela permite a construção de textos a partir de outros textos, de forma a se aproximar da abordagem humana.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são descritas técnicas e conceitos utilizados nos artigos selecionados, além das arquiteturas propostas nas cinco publicações estudadas.

2.1 REDE NEURAL ARTIFICIAL

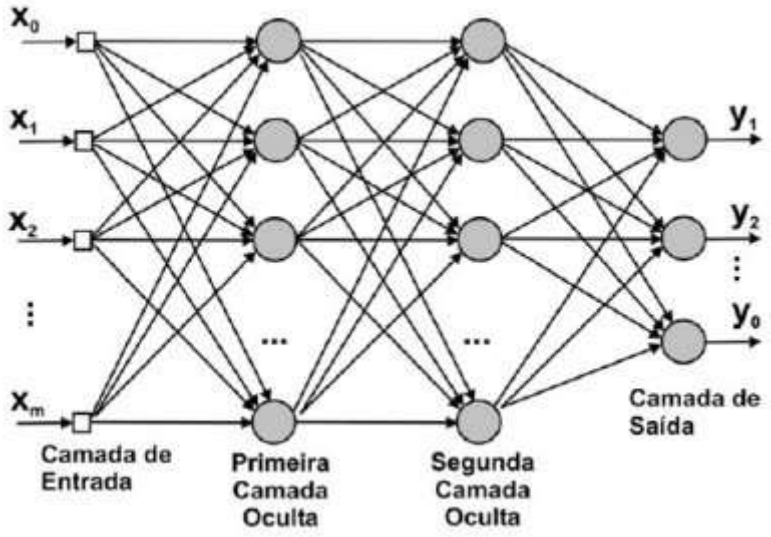


Figura 1: Rede neural artificial.

Fonte: Barbosa et al, 2005.

Buscando assemelhar o comportamento do sistema nervoso central de seres vivos, redes neurais artificiais têm como objetivo generalizar características gerais das entradas para predizer as saídas esperadas. Para isso, redes neurais artificiais são formadas por neurônios (círculos cinzas na figura acima).

Os neurônios são responsáveis pelo aprendizado de uma rede neural artificial. Eles recebem entradas e as multiplicam por matrizes denominadas pesos, que são responsáveis pelo aprendizado da rede. Para isso, é gerada uma saída para cada entrada, que é comparada com a saída esperada. A diferença entre as duas é utilizada para atualizar os pesos de cada neurônio a depender do quanto cada um contribuiu para a saída final.

Essa atualização de pesos é denominada retropropagação, pois o erro final é retropropagado para as camadas internas da rede. Uma rede neural pode ser definida

como um conjunto de neurônios organizados em camadas de neurônios, que podem ou não estar conectados a todos outros neurônios das camadas posterior e inferior.

2.2 REDE NEURAL RECORRENTE

Rede neural recorrente é projetada para reconhecer padrões em sequências de dados como: texto, escrita à mão, séries numéricas, etc. Diferentemente de redes progressivas, na qual os dados de entrada percorrem as camadas em apenas uma direção, sem conexões entre neurônios de uma mesma camada, redes neurais recorrentes podem ser retroalimentadas pelos dados de saída, permitindo a memorização de entradas anteriores. Esse tipo de característica é de grande importância em uma sequência de entrada muito grande e com dependência entre os dados.

Uma rede neural recorrente utiliza os dados da entrada atual e anterior para produzir a saída atual. Desse modo, ocorre uma forma de memorização dos dados já utilizados no treinamento, o que possibilita que a rede se beneficie do conhecimento sobre a sequência de entrada durante a geração da saída desejada. Para prover esta característica, RNNs (*Recurrent Neural Network*) utilizam um *loop* simples, na qual as informações da iteração anterior são agregadas às informações da iteração atual.

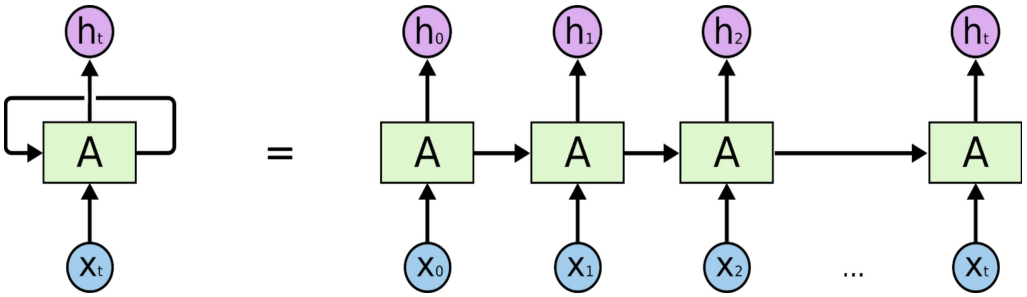


Figura 2: Rede neural recorrente.

Fonte: Christopher Olah, 2015.

Na figura acima, cada neurônio da RNN tem duas entradas: a entrada da etapa de tempo atual e a saída gerada na etapa de tempo anterior. Este tipo de rede é adequado para o tratamento de sequências e listas, o que propicia o uso de informações semânticas pela RNN, caracterizada como uma rede neural com memória. Por esta razão, RNNs são utilizadas para várias tarefas na qual o contexto é determinante para uma boa

abstração de padrões, tais como: classificação de imagens, legenda de imagens, tradução de textos em uma dada língua, análise de sentimentos e sumarização de textos (foco deste trabalho).

Para a atualização dos pesos é utilizada uma técnica chamada retropropagação no tempo, na qual a entrada é composta de uma sequência completa, e o erro final é a soma de todos os erros de cada etapa de tempo. Logo, é necessário calcular o erro em cada etapa de tempo e atualizar os pesos, o que é um processo lento devido ao alto custo computacional (Ilya Sutskever, 2013).

Entretanto, embora sejam melhores que as redes neurais tradicionais, RNNs sofrem com um problema conhecido como dissipação do gradiente (*vanishing gradient*), na qual o gradiente vai diminuindo a cada etapa até chegar a um valor insignificante para o ajuste dos pesos. Essa deficiência limita a utilização da RNN a uma memória de curto prazo, o que inviabiliza sua utilização em aplicações que necessitam guardar informações de entradas anteriores por um longo prazo. Duas alternativas para tratar esta questão são: LSTM (*Long Short Term Memory*) e GRU (*Gated Recurrent Unit*).

2.2.1 *Long Short Term Memory* (LSTM)

LSTM é utilizada pela sua capacidade de aprender dependências de longo prazo com um desempenho significativamente superior em relação às RNNs tradicionais, além de ser efetiva para entradas com intervalos de tempo de duração desconhecida. Sua arquitetura é composta de três portões, redes neurais internas responsáveis pelo processamento das informações, e estados das células, também chamadas de blocos de memória, que são responsáveis pela retenção das informações de longo prazo (Sundermeyer et al., 2012).

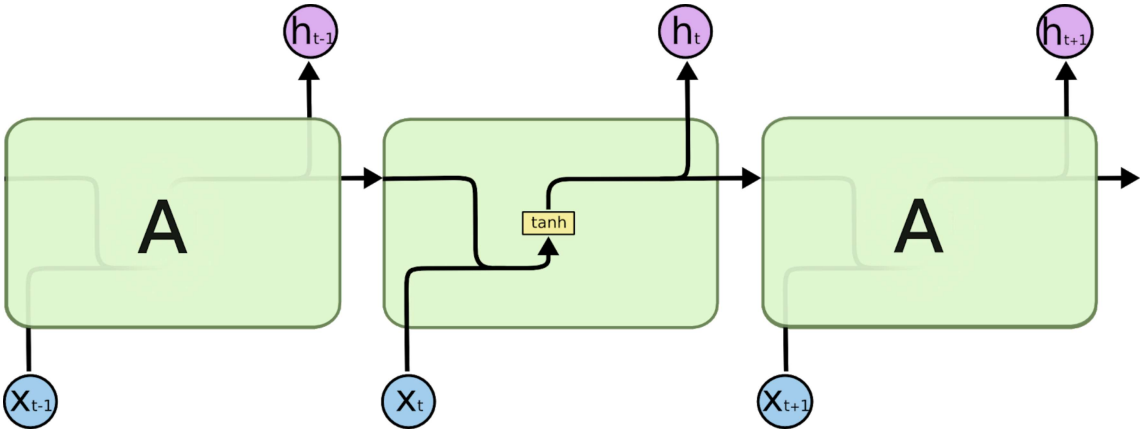


Figura 3: Estrutura de uma RNN padrão.

Fonte: Christopher Olah, 2015.

Uma RNN padrão é composta por uma cadeia de módulos de redes neurais repetidas, com uma estrutura simples, na qual o conteúdo da célula sempre é substituído a cada etapa de tempo, utilizando o estado oculto anterior e a entrada atual. Na figura 3, o módulo é formado por uma camada única de tangente hiperbólica, que ajusta os valores entre -1 e 1.

LSTMs também contém uma estrutura de cadeia, mas com 4 camadas de rede neural, os portões citados no parágrafo anterior. Diferentemente de RNNs, o conteúdo da célula é atualizado adicionando valores da entrada atual, sem substituir os valores anteriores. Essa abordagem de adição permite que o conteúdo anterior seja mantido, apenas adicionando novas informações no topo (Chung et al., 2014).

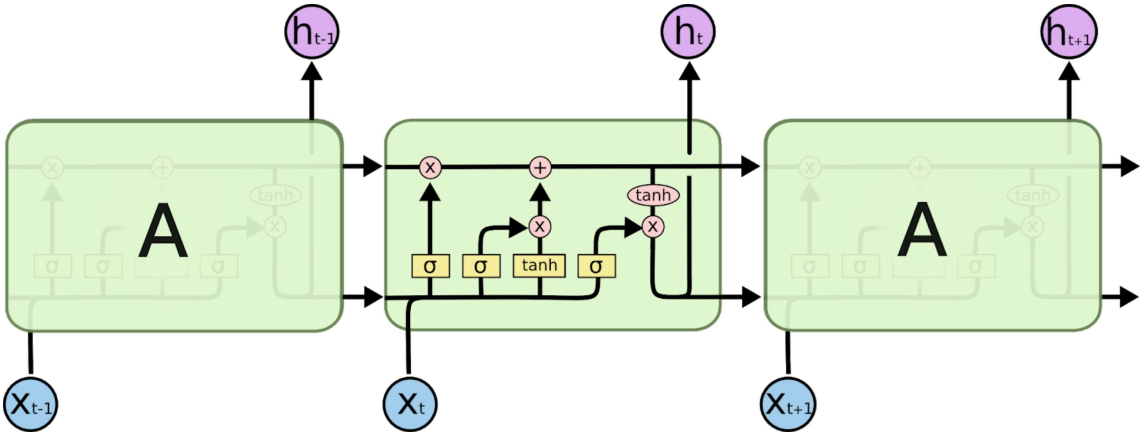
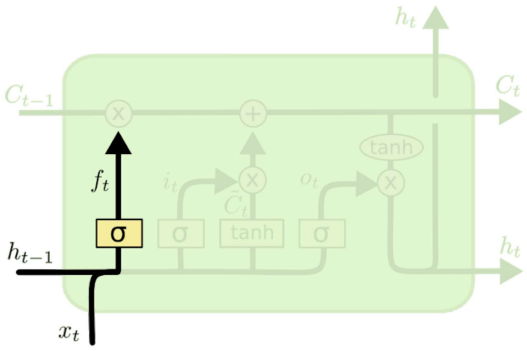


Figura 4: Estrutura de uma célula LSTM.

Fonte: Fonte: Christopher Olah, 2015.

A figura 4 representa a estrutura de uma LSTM, formada por 3 portões, o estado da célula (linha horizontal superior) e o estado oculto. Cada portão interage com o estado da célula de forma única, controlando o fluxo de informações pela rede. O estado da célula transmite as informações por toda a rede, podendo ser interpretado como uma memória de longo prazo. Em RNNs, apenas o estado oculto é utilizado, podendo ser considerado uma memória de curto prazo, pela dificuldade em manter informações na memória em situações na qual a entrada é muito longa.

Em uma LSTM, há três portões com camadas sigmóides, que permitem ou bloqueiam a passagem de informações utilizando multiplicação elemento por elemento. Uma camada sigmóide gera saídas entre 0, na qual nada é transmitido, e 1, na qual tudo é transmitido.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

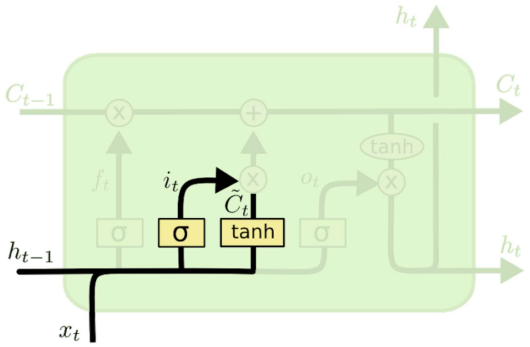
Figura 5: Portão de esquecimento.

Fonte: Christopher Olah, 2015.

Como entrada, uma célula LSTM recebe a entrada atual x_t, o estado oculto anterior (memória de curto prazo) h_{t-1} e o estado da célula anterior (memória de longo prazo) C_t. Dessa forma, o estado da célula representa a memória seletiva da rede, enquanto o estado oculto representa a memória de tudo que já foi visto até o momento. A primeira etapa do funcionamento de uma LSTM é suportada pelo portão de esquecimento, no qual é calculado o quanto da informação do estado da célula deve ser mantida.

O portão de esquecimento recebe duas entradas, x_t e h_{t-1}, que são multiplicadas pela matriz de peso W_f e somadas ao viés b_f, que é utilizado para uma melhor adaptação da rede aos dados fornecidos, movendo a função de ativação para esquerda ou direita, o que pode ser crítico para o sucesso no aprendizado. O resultado serve como

entrada para a camada sigmóide, que gera saídas entre 0 e 1, esquecendo ou mantendo a informação.

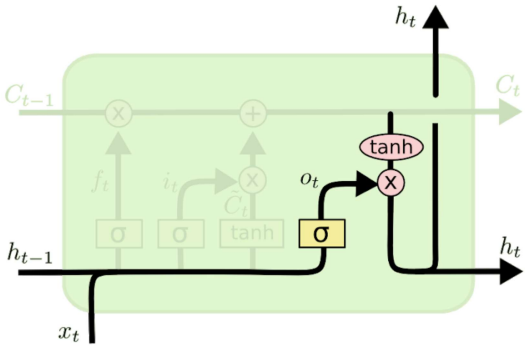


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 6: Portão de entrada.

Fonte: Christopher Olah, 2015.

A próxima etapa é suportada pelo portão de entrada, no qual é calculado o quanto das novas informações serão adicionadas ao estado da célula. Primeiro, uma camada sigmóide calcula o quanto os valores serão atualizados utilizando as entradas x_t e h_{t-1} , de forma semelhante às operações do portão de esquecimento: multiplicando as duas entradas pela matriz de pesos W_i e somando o resultado com viés b_i. Depois uma camada de tangente hiperbólica gera um vetor $\tilde{C}_t$ de novos valores entre -1 a +1, candidatos a entrar no estado da célula, utilizando as mesmas entradas e operações da camada sigmóide. Em seguida, os valores de $\tilde{C}_t$ e a saída da camada sigmóide são multiplicados elemento a elemento, para gerar as informações que serão adicionadas ao estado da célula.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figura 7: Portão de saída.

Fonte: Christopher Olah, 2015.

Para a última etapa, a decisão de qual será a saída da LSTM, o portão de saída é o responsável. Para isso, as informações do estado da célula atual são geradas ao multiplicar duas saídas elemento por elemento: o_t, saída da camada sigmóide, utilizada para decidir os valores que serão lembrados utilizando as entradas x_t e h_{t-1}, multiplicadas pelo peso W_o, com o resultado somado ao viés b_o; e da camada de tangente hiperbólica, que gera saídas entre -1 e 1, tendo o estado da célula atual como entrada. Consequentemente, a camada sigmóide é responsável pelo cálculo do quanto das informações da célula farão parte da saída h_t da LSTM.

2.2.2 *Gated Recurrent Unit* (GRU)

GRU carrega grandes semelhanças com LSTM. Basicamente as diferenças são: (a) utilização de apenas dois portões, portão de atualização e portão de redefinição, responsáveis pelo cálculo de quanto das informações devem ser mantidas ou descartadas; e (b) utilização do estado oculto para transmitir as informações de memória de curto e longo prazo, descartando a necessidade do estado de célula. Logo, utiliza menos parâmetros, necessitando de menos recursos, sendo mais simples para treinar. Contudo, para conjuntos de dados com sequências muito longas, é menos precisa que LSTM.

A escolha entre os dois tipos de redes neurais recorrentes depende fortemente do conjunto de dados e do problema trabalhado[1]. Por exemplo, em Britz et al. (2017), LSTM superou GRU consistentemente em experimentos com modelo *encoder-decoder*. Todavia, em Kaiser et al. (2015), GRU superou LSTM em todas as tarefas testadas, com exceção da modelagem de linguagem.

[1] Chung et al., 2014.

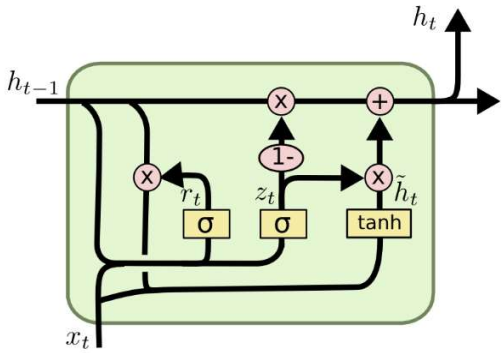
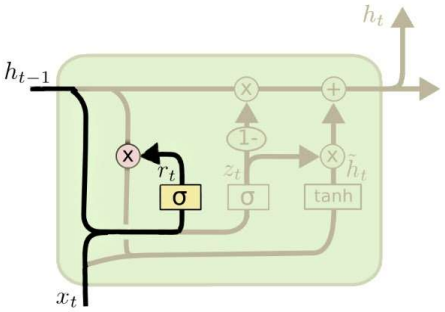


Figura 8: Estrutura de uma GRU.

Fonte: Christopher Olah, 2015.

A figura 8 representa a estrutura de uma célula GRU, composta por três camadas de redes neurais que formam os dois portões citados no parágrafo anterior: atualização e redefinição. Assim como na estrutura LSTM, os portões têm a função de filtrar o quanto das informações serão mantidas na memória para uso futuro. Para isso, cada célula GRU recebe como entrada o estado oculto anterior h_{t-1} e a entrada atual x_t.

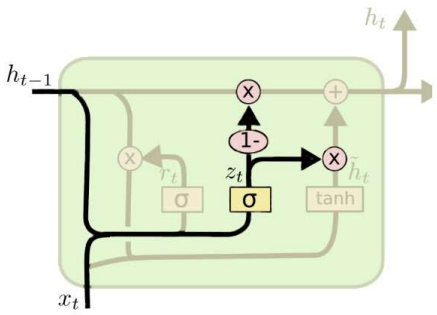


$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

Figura 9: Portão de redefinição.

Fonte: Christopher Olah, 2015.

Para a primeira etapa, o portão de redefinição recebe as duas entradas da célula, h_{t-1} e x_t, multiplica-as por W_r, matriz de pesos do portão de atualização, e soma os resultados antes de passar por uma camada sigmóide, que ajusta os valores entre 0 e 1. A saída da camada sigmóide é multiplicada elemento a elemento com o estado oculto anterior h_{t-1}, gerando o vetor r_t como saída. Finalmente, o portão calcula o quanto das informações repassadas pela etapa de tempo anterior serão mantidas.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figura 10: Portão de atualização.

Fonte: Christopher Olah, 2015.

Na segunda e última etapa, é utilizado o portão de atualização, responsável por calcular o quanto das novas informações serão adicionadas ao estado oculto atual h_t. Assim como o portão de redefinição, o portão de atualização recebe como entradas h_{t-1} e x_t, multiplicando-as por uma matriz de pesos, W_z. O resultado serve como entrada para uma camada sigmóide, gerando o vetor z_t como saída, que será utilizado para a obtenção da saída final da célula.

A entrada atual x_t é concatenada com o resultado da multiplicação elemento por elemento da saída do portão de redefinição r_t e o estado oculto da etapa de tempo anterior h_{t-1}, servindo de entrada para uma camada de tangente hiperbólica que gera o vetor $\tilde{h}_t$, que será multiplicado elemento por elemento com z_t. A saída final da célula, o estado oculto h_t, é obtida ao somar os resultados de duas multiplicações: $1 - z_t$ multiplicado pelo estado oculto da célula anterior h_{t-1}; e z_t multiplicado por $\tilde{h}_t$. Caso o valor de z_t seja 1, a informação em h_{t-1} será substituída por $\tilde{h}_t$. Caso seja 0, a informação será mantida na memória.

2.2.3 *Encoder-decoder Sequence to Sequence*

Um tipo de rede neural muito utilizado na área de tradução e reconhecimento de fala é o modelo *sequence to sequence*, com ótimos resultados (Sutskever et al., 2014; Venugopalan et al., 2015; Prabhavalkar et al., 2017). O modelo foi introduzido para lidar com problemas no qual os tamanhos da entrada e saída podem ser diferentes.

Consiste em duas RNNs, *encoder* e *decoder*, na qual a primeira mapeia uma sequência de entrada de tamanho variável em um vetor de representação de tamanho fixo, e a segunda mapeia o vetor de representação em uma sequência de saída de tamanho variável (Cho et al., 2014). Esse tipo de situação é frequente em traduções, o que explica a grande utilização do modelo para esse problema.

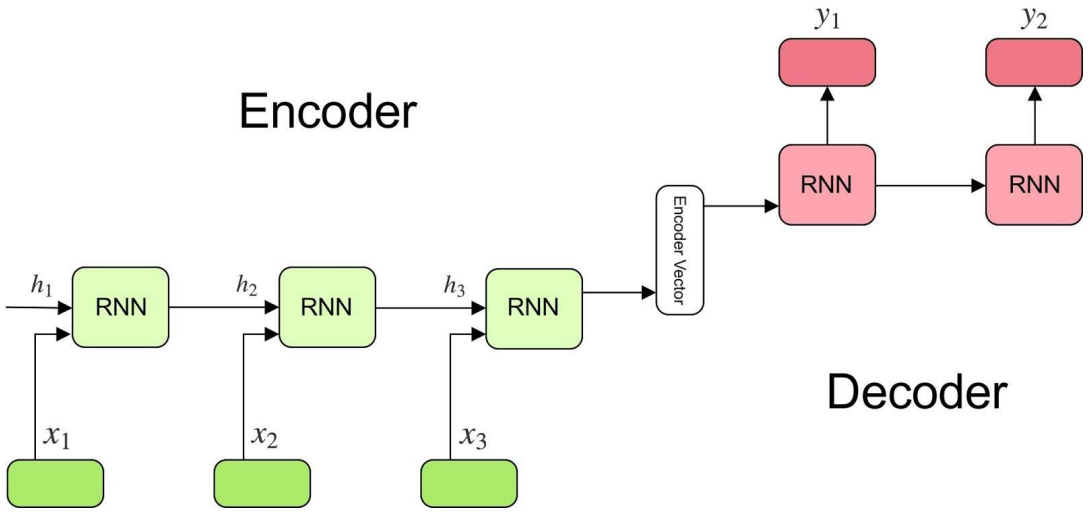


Figura 11: Estrutura de um modelo *encoder-decoder* básico.

Fonte: Simeon Kostadinov, 2019.

Na figura acima, temos um modelo *encoder-decoder*, que é composto por três partes: *encoder*, vetor *encoder* e *decoder*. Cada parte tem uma função específica na estrutura do modelo. O *encoder* gera um vetor de representação da sequência de entrada, de tamanho fixo (*Encoder vector* na figura), que é utilizado pelo *decoder* para gerar a sequência de saída.

O *encoder* é formado por redes neurais recorrentes, geralmente LSTMs ou GRUs, responsáveis pelo processamento das entradas do modelo. Nesta etapa, a RNN do *encoder* recebe uma palavra da sequência de entrada junto do estado oculto anterior, como citado anteriormente. Ao final, tem um vetor de representação como saída, utilizado pelo *decoder* para gerar as saídas finais da rede.

O vetor do *encoder* é o estado oculto produzido pelo *encoder*. Ou seja, o estado oculto da última etapa de tempo do *encoder*, sendo utilizado como o estado oculto

inicial do *decoder*. É responsável pela captura as informações de todos os elementos da entrada, auxiliando o *decoder* a produzir saídas consideradas corretas.

O *decoder* assim como o *encoder,* é composto de redes neurais recorrentes, geralmente LSTMs ou GRUs, na qual a célula recebe o estado oculto etapa de tempo anterior e gera uma saída e o estado oculto da etapa de tempo atual, que servirá como entrada para a célula na etapa de tempo posterior.

2.2.3.1 Busca em feixe e busca gulosa

Gerar a sequência de saída utilizando a distribuição de vocabulário não é uma tarefa trivial. Para determinada entrada, há uma ampla possibilidade de saídas possíveis. Sendo assim, a decisão sobre qual saída é a melhor entre todas as possíveis pode ser abordada de várias formas, sendo a busca em feixe a solução mais utilizada.

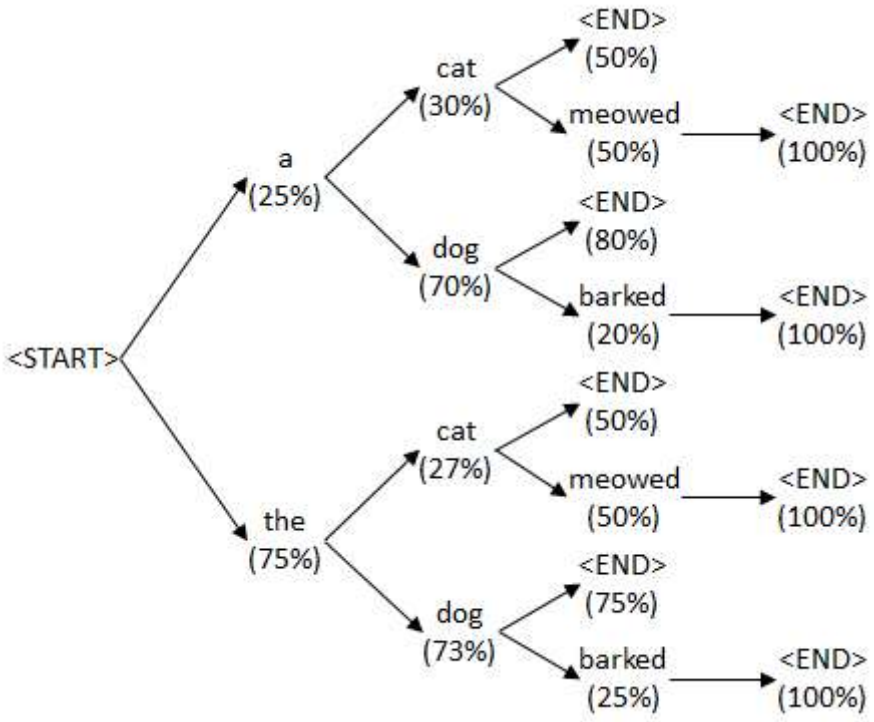


Figura 12: Busca em feixe da sentença de saída.

Fonte: Geeky is Awesome., 2016.

A busca em feixe trabalha com um número limitado de possíveis sentenças de saídas (hipóteses). A cada etapa do *decoder*, o algoritmo calcula as K sequências com maior probabilidade final (produtos das probabilidades de cada uma das palavras que a

compõem). Por exemplo, no caso da figura acima, na qual K é igual a dois,, o algoritmo selecionará as duas palavras com maior probabilidade na distribuição de vocabulário (saída do *decoder*). A cada etapa, apenas as duas sentenças com maior probabilidade continuam como hipóteses.

A busca gulosa é uma busca em feixe com apenas uma hipótese. Logo, para cada etapa do *decoder*, o algoritmo compara cada palavra da distribuição do vocabulário com a sequência com maior probabilidade final da etapa anterior. Embora consuma menos recursos, essa solução gera resultados piores que a busca em feixe (Chopa et al., 2016). Por exemplo, no caso da figura acima, caso fosse utilizada busca gulosa, a saída gerada seria "the dog", o que não é uma sentença correta no contexto apresentado na imagem.

2.2.4 Mecanismos de atenção

Um problema comum de redes neurais recorrentes, já citado anteriormente, é a dissipação do gradiente. As soluções já citadas, LSTM e GRU, conseguem amenizar o problema até certo ponto. Porém, para sequências muito longas, LSTMs e GRUs não são suficientes para evitar o problema de forma satisfatória (Bahdanau et al., 2015).

Para tratar essas deficiências, mecanismos de atenção foram introduzidos, e têm sido utilizados amplamente por trabalhos na área. Com sua utilização, o *decoder* aprende a focar em partes relevantes da entrada em cada etapa da geração da saída, dependendo da sentença de entrada e do que já foi gerado até o momento. Desta forma, o *decoder* não depende do estado oculto final do *encoder* para as predições. Esta dependência acarreta em problemas quando as sentenças de entrada são muito longas.

Em um modelo com mecanismo de atenção, ocorrerá o aprendizado de um alinhamento entre a sequência de entrada e a geração da saída. No exemplo da figura 13, para cada estado oculto da entrada (h_t), é calculado uma pontuação de atenção ($a_{t,i}$). O vetor de atenção (a_t) é formado por todas pontuações calculadas. Assim sendo, o modelo considerará um peso para cada palavra da sequência de entrada durante a etapa de geração de saída.

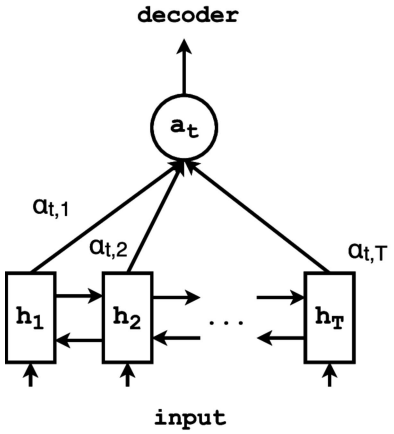


Figura 13: Modelo *encoder-decoder* com mecanismo de atenção.

Fonte: Kann et al., 2016.

Desta forma, utiliza apenas as informações mais relevantes para a predição. Por exemplo, para gerar o vetor de contexto c_t, a seguinte equação é utilizada:

$$c_t = \sum_{n}^{i=1} \alpha_{ti} h_i$$

Na qual α_{ti} representa o alinhamento entre a saída y_t e a entrada x_i. O valor de α_{ti} é calculado utilizando as seguintes equações:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ti} = a(s_{i-1}, h_t).$$

Na qual e_{ti} é uma pontuação que representa o quão bem a entrada na posição t e a saída na posição i combinam. Assim sendo, o mecanismo de atenção define o quanto de cada estado oculto h_i deve ser considerado para o contexto da saída y_t.

Além disso, é possível visualizar a matriz de alinhamento do mecanismo de atenção de forma a entender os pesos das relações entre a entrada e a saída do modelo. Na figura 14, por exemplo, é representado o alinhamento entre a entrada, uma frase em francês, e a saída, a tradução em inglês. Quanto mais clara for a posição de intersecção, maior é o peso entre as duas palavras.

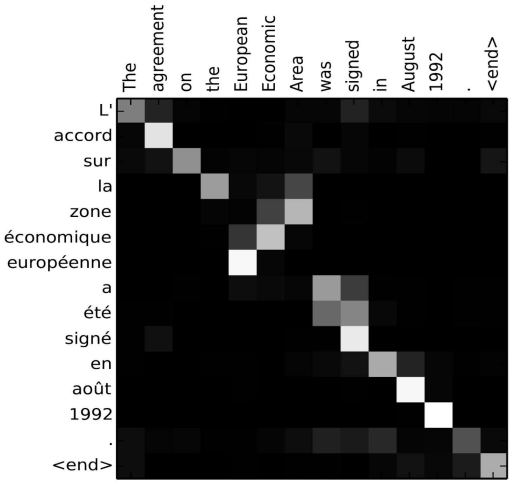


Figura 14: Matriz de alinhamento: frase em francês e sua tradução em inglês.

Fonte: Bahdanau et al., 2015.

2.2.4.1 Intra-atenção

Um dos mecanismos de atenção que tem sido utilizado com sucesso na área de sumarização abstrativa é o intra-atenção. Nele, o modelo é treinado para aprender as relações entre diferentes posições de uma sequência de entrada, utilizando-as para gerar representações dessa mesma sequência. Dessa forma, as correlações entre as palavras atuais e as palavras anteriores da sequência são utilizadas pelo *decoder* no momento de predição da saída.

A intra-atenção tem mostrado capacidade para aprender padrões sintáticos complexos como, por exemplo, sentenças compostas de frases coordenativas e subordinativas. Por exemplo, Transformers (Vaswani, 2017), modelo que utiliza apenas intra-atenção, sem RNNs, conseguiu alcançar resultados de estado da arte sendo mais rápido para treinar e paralelizável, além de aprender relações sintáticas sofisticadas durante o treinamento.

2.2.4.2 Atenção global e atenção local

Os mecanismos de atenção podem ser divididos em dois tipos: global e local. Ambos atuam no *decoder*, sendo diferenciados pela forma como o vetor de contexto c_t é calculado. Na atenção global, todas as posições da entrada são utilizadas no cálculo, enquanto a atenção local foca em uma parte específica.

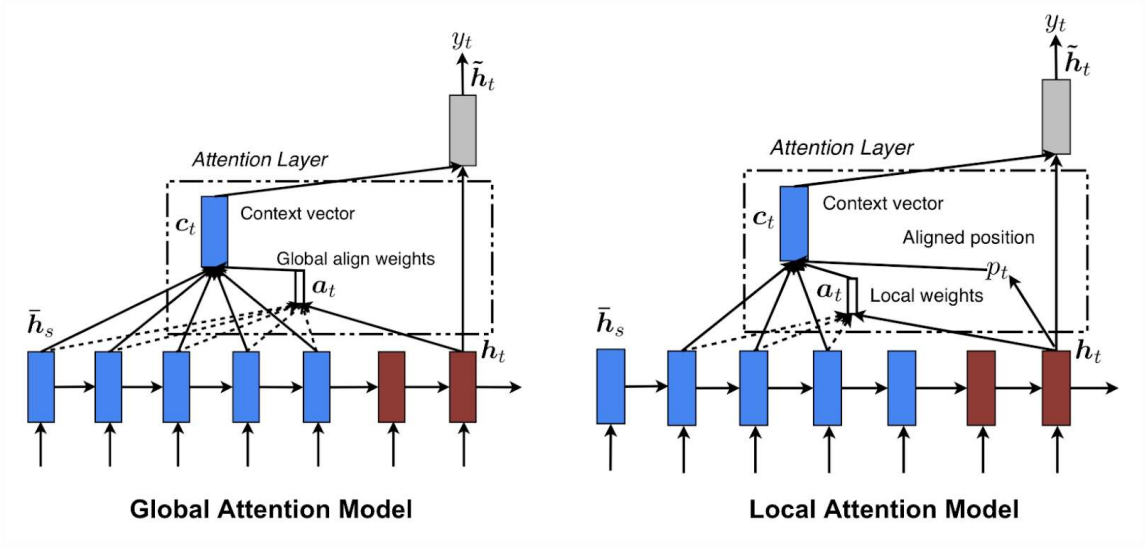


Figura 15: Estrutura de um modelo NMT (Neural Machine Translation) com atenção global e atenção local.

Fonte: Luong et al., 2015.

Na figura 15, é representado um modelo com atenção global e um modelo com atenção local, no qual é possível visualizar o vetor de alinhamento a_t, utilizado para ponderar o peso que cada palavra tem no contexto da sentença. Na atenção global, a_t é calculada levando em consideração toda os estados ocultos da sequência de entrada para toda palavra da saída.

Diferentemente da atenção global, a atenção local não considera todos os estados ocultos da sequência de entrada, devido ao alto custo computacional, focando apenas em uma parte dos estados ocultos para cada palavra da saída gerada. Para isso, além de a_t, o modelo com atenção local gera uma posição alinhada p_t para a saída y_t, que é utilizada para calcular o vetor de contexto c_t como uma média ponderada dos estados ocultos que estiverem dentro de uma janela $[p_t - D, p_t + D]$, na qual D é um valor escolhido a depender do problema trabalhado.

2.3 SUMARIZAÇÃO ABSTRATIVA

No contexto de Processamento de Linguagem Natural, no qual busca-se gerar e compreender de maneira automática as línguas humanas naturais, há várias aplicações, tais como: máquina de tradução, análise de discurso, respostas a perguntas, entre outras. Uma das principais é a sumarização automática de textos, que pode ser abordada de

forma extrativa, gerando um subconjunto da sentença de entrada, ou abstrativa, generalizando a entrada e gerando a sentença de saída a partir da generalização, não necessariamente sendo um subconjunto da entrada. A abordagem abstrativa é focada neste trabalho, sendo explicada a seguir.

Dada uma sequência de entrada consistindo de M palavras $x_1, ..., x_M$, derivadas do vocabulário fixo ν de tamanho $|\nu| = V$, cada palavra é representada por um vetor $x_i \in \{0,1\}^V$, no qual $i \in \{1, ..., M\}$, sentenças são sequências de representações, X é o conjunto de possíveis entradas e $X_{[i,j,k]}$ representa uma subsequência de elementos i, j, k.

Um gerador de sumários recebe x como entrada e gera como saída uma sentença y de tamanho $N < M$, na qual as palavras do sumário são do mesmo vocabulário υ, e a saída é uma sequência $y_1, \cdots, y_N$, na qual N é fixo (o modelo sabe o tamanho do sumário antes de gerá-lo). Para definir o problema de geração de sumários, é assumido que $\mathcal{Y} \subset \left(\{0,1\}^V, ..., \{0,1\}^V\right)$ representa todas as possíveis sentenças de tamanho N, na qual para cada i e $y \in \mathcal{Y}$, y_i é uma representação. Define-se um sistema como abstrativo se o mesmo tenta encontrar a sequência ótima desse conjunto $\mathcal{Y}$:

$$\begin{aligned} &\arg\max s(\boldsymbol{x}, \boldsymbol{y}), \\ &y \in \mathcal{Y} \end{aligned} \tag{1}$$

na qual a função de pontuação pode ser definida como $s : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$. Um modelo abstrativo pode gerar novas sentenças durante a geração do sumário. Já um sistema totalmente extrativo, no qual os sumários são gerados transferindo palavras da entrada para a saída, sem adicionar palavras não encontradas na entrada, pode ser representado como:

$$\begin{aligned} &\arg\max s(x, x_{[m_1,...,m_N]}), \\ &m \in \{1, ..., M\}^N \end{aligned}$$

2.3 Artigos selecionados

O primeiro artigo propõe um modelo de sumarização abstrativa que utiliza atenção baseada em Bahdanau et al. (2014) para gerar a sentença de saída (sumário). O

modelo também utiliza busca em feixe no decoder, com o sumário de saída contendo apenas uma sentença.

Já o segundo artigo propõe um modelo que também utiliza atenção e também gera sumários com uma sentença, assim como o artigo 1. Entretanto, ao invés de uma rede progressiva, utiliza uma rede convolucional no encoder. O modelo do segundo artigo consegue superar o desempenho do modelo do primeiro artigo nas métricas testadas (ROUGE).

O terceiro artigo propõe alguns modelos que utilizam atenção, características linguísticas, gerador de ponteiro para tratamento de palavras desconhecidas (fora do vocabulário fixo) e uma nova base de treinamento para sumarização abstrativa de múltiplas sentenças. Os resultados na métrica ROUGE superam os modelos do artigo 1 e 2. Além disso, outra vantagem é que o modelo gera sumários com mais de uma sentença e copia palavras desconhecidas da entrada para a saída, o que aumenta a legibilidade dos sumários gerados.

O quarto artigo propõe um modelo que também utiliza atenção e geração de ponteiro para tratamento de palavras desconhecidas, combinados com um vetor responsável por memorizar a cobertura da entrada até a etapa atual de geração do decoder, para tratamento de repetição de frases. Esse vetor, denominado vetor de cobertura, é a soma das atenções das etapas anteriores do decoder. Utilizando a base de treinamento proposta pelo artigo 3, o modelo do artigo 4 gera sumários compostos por mais de uma sentença. Os resultados na métrica ROUGE do artigo 4 superam os modelos apresentados nos três artigos anteriores.

Por último, o quinto artigo propõe um modelo com atenção tanto no encoder como no decoder (artigos anteriores utilizavam apenas no encoder), geração de ponteiro para tratamento de palavras desconhecidas, uma nova função objetivo que combina aprendizado supervisionado e aprendizado por reforço, uma única matriz de representação para o encoder e decoder, diminuindo os parâmetros para treinamento. Da mesma forma que os artigos 4 e 3, os sumários gerados são compostos por múltiplas sentenças. Os resultados dos testes do modelo do artigo 5 superam os 4 artigos anteriores.

2.4 Artigo 4 - Get To The Point: Summarization With Pointer-generator Networks

No artigo 4 é proposto o uso de um gerador de ponteiros para tratar palavras desconhecidas e atenção combinado com um vetor de cobertura, responsável por informar ao modelo quais palavras da entrada já foram focadas em etapas anteriores do decoder. O processo de sumarização do modelo do artigo 4 pode ser descrito com as seguintes etapas:

- Encoder
- Decoder
- Atenção com vetor de cobertura
- Vetor de contexto
- Distribuição de vocabulário
- Geração de ponteiro
- Distribuição final

A seguir, cada etapa é detalhada.

2.4.1 Encoder

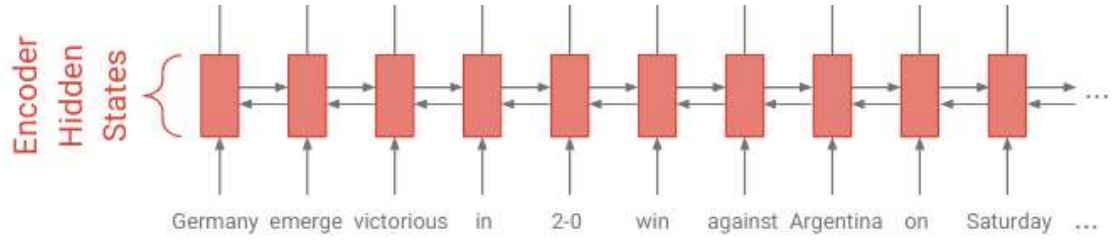


Figura 16: Encoder do modelo do artigo 4.

Fonte: See et al., 2017.

Responsável pela entrada do modelo, o encoder ilustrado acima recebe a sentença de entrada palavra por palavra, nas duas direções (ordem normal e reversa), calculando os estados ocultos de cada uma delas. O último estado oculto do encoder representa a sentença de entrada como um todo, sendo uma das entradas do decoder na primeira etapa de geração.

2.4.2 Decoder

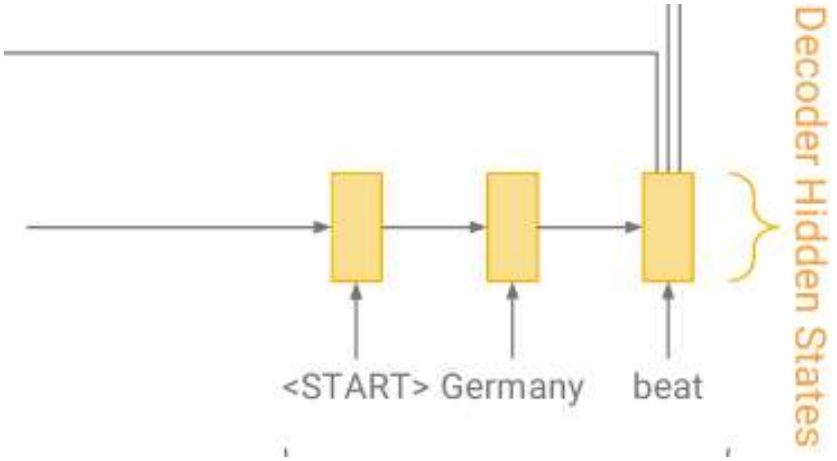


Figura 17: *Decoder* do modelo do artigo 4.

Fonte: See et al., 2017.

Após o encoder receber toda a sentença de entrada, o último estado oculto é repassado ao decoder, que recebe a tag <START>, indicando o início da geração da sentença de saída, calcula o estado oculto que será utilizado para o cálculo da atenção, probabilidade de geração de ponteiro e distribuição de vocabulário.

2.4.3 Atenção com vetor de cobertura

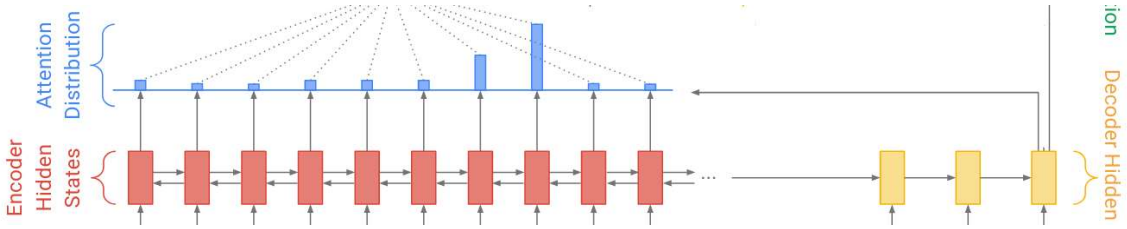


Figura 18: Atenção do modelo do artigo 4.

Fonte: See et al., 2017.

Como ilustrado na figura 18, assim como nos artigos anteriores, é calculado a pontuação de atenção de cada palavra da entrada utilizando o estado oculto da etapa do decoder e os estados ocultos do encoder. Entretanto, o modelo do artigo 4 incorpora um vetor de cobertura, representando as palavras já focadas nas etapas anteriores do decoder, sendo composto pelas somas das atenções anteriores.

2.4.4 Vetor de contexto

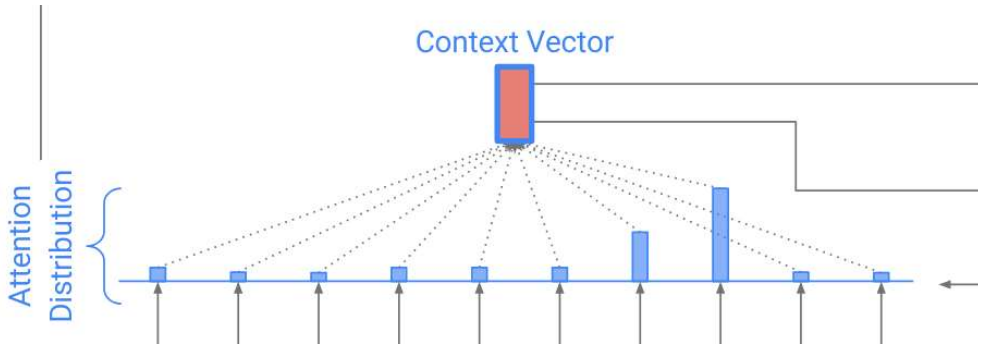


Figura 19: Vetor de contexto do modelo do artigo 4.

Fonte: See et al., 2017.

O vetor de contexto é responsável por representar o contexto da entrada para a etapa atual do decoder. Utilizando as pontuações calculadas na etapa anterior, os estados ocultos do encoder são ponderados, gerando como resultado o vetor de contexto. O vetor de contexto é utilizado no cálculo da probabilidade de geração de ponteiro e a distribuição de vocabulário.

2.4.5 Distribuição de vocabulário

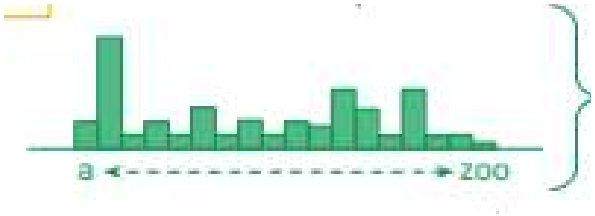


Figura 20: Distribuição de vocabulário do modelo do artigo 4.

Fonte: See et al., 2017.

A distribuição de vocabulário, ilustrada na figura acima, é uma distribuição de probabilidades com o mesmo tamanho do vocabulário fixo (vocabulário utilizado pelo decoder). Cada posição indica a probabilidade da palavra encontrada na mesma posição do vocabulário fixo ser gerada na etapa atual do decoder. É calculada utilizando o vetor de contexto e o estado oculto do decoder.

2.4.6 Geração de ponteiro

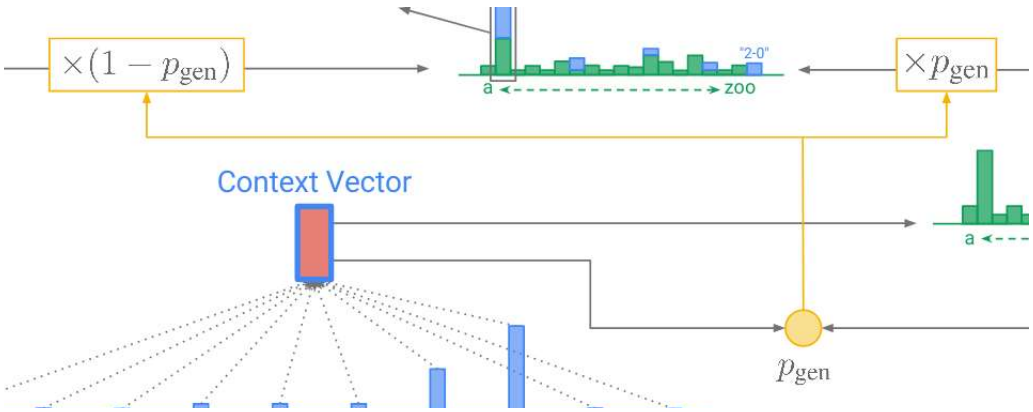


Figura 21: Gerador de ponteiro do modelo do artigo 4.

Fonte: See et al., 2017.

Responsável pelo tratamento de palavras desconhecidas, o gerador de ponteiro aponta para o estado oculto do encoder com a maior pontuação de atenção para a etapa atual do decoder e copia a palavra daquela posição na entrada para a sentença de saída. É uma probabilidade calculada utilizando o estado oculto do decoder e o vetor de contexto. A figura 21 demonstra como o gerador de ponteiro é responsável por indicar se o decoder deve utilizar a distribuição de vocabulário ou a atenção para gerar a próxima palavra da sentença de saída.

2.4.7 Distribuição final

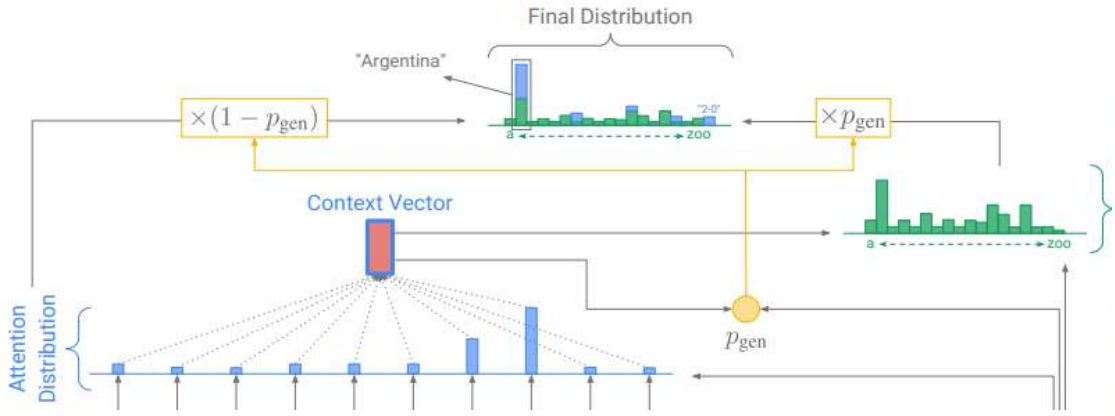


Figura 22: Distribuição final do modelo do artigo 4.

Fonte: See et al., 2017.

A distribuição final do modelo pode ser representada pela combinação da distribuição de vocabulário e a distribuição de atenção, já que a saída pode ser gerada tanto utilizando uma ou outra. A figura 22 demonstra isso sobrepondo a distribuição de vocabulário com as pontuações das palavras encontradas na distribuição de atenção.

2.4.8 Modelo do artigo 4

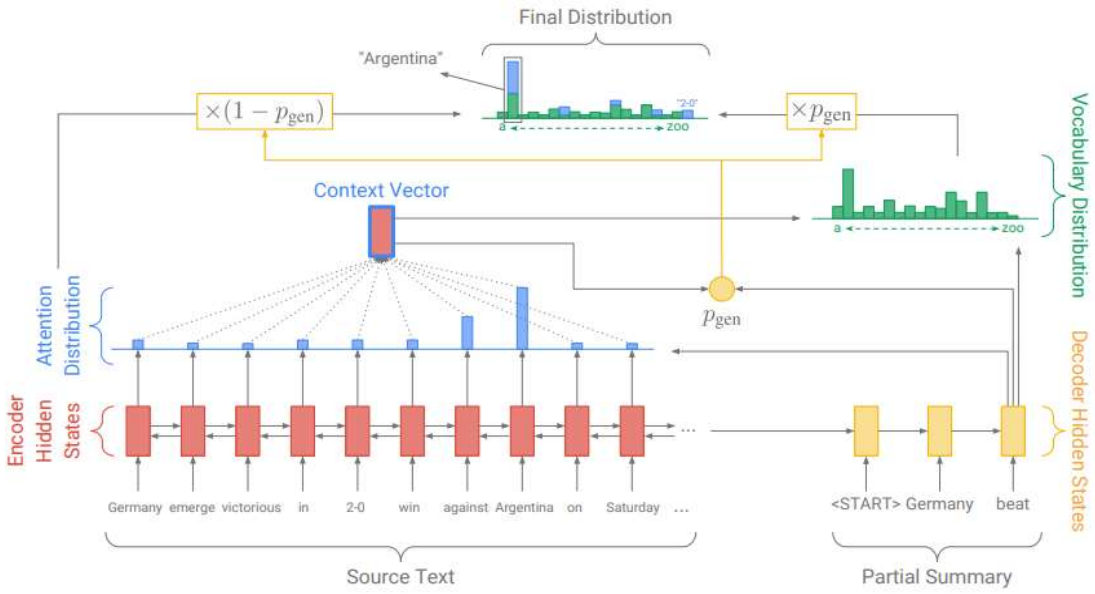


Figura 23: Arquitetura completa do modelo do artigo 4.

Fonte: See et al., 2017.

Acima, na figura 23, temos o modelo completo do artigo 4. O encoder bidirecional no canto inferior esquerdo, recebendo as palavras de entrada e com o último estado oculto servindo de entrada para a primeira etapa do decoder, que também recebe a tag <START> para iniciar a geração da saída.

O decoder calcula o estado oculto que é utilizado para calcular a pontuação de atenção de cada estado oculto do encoder. A atenção é utilizada para ponderar cada estado oculto do encoder, gerando o vetor de contexto, utilizado junto do estado oculto do decoder para calcular a distribuição de vocabulário e a probabilidade do gerador de ponteiro.

2.5 Artigo 5 - Deep Reinforced Model For Abstractive Summarization

No artigo 5 é proposto um modelo de sumarização abstrativa que utiliza atenção e gerador de ponteiro. Também é proposto uma nova função objetivo, que combina aprendizado supervisionado e aprendizado por reforço, além do uso de uma matriz de representação única para o encoder e o decoder. O processo de sumarização do modelo do artigo 5 é semelhante ao modelo do artigo 4. Por essa razão, apenas as diferenças serão comentadas nas subseções posteriores. O processo é composto das seguintes etapas:

- Encoder
- Decoder
- Atenção no encoder e decoder
- Vetores de contexto
- Distribuição de vocabulário
- Geração de ponteiro
- Distribuição final

2.5.1 Atenção intra-decoder e vetores de contexto

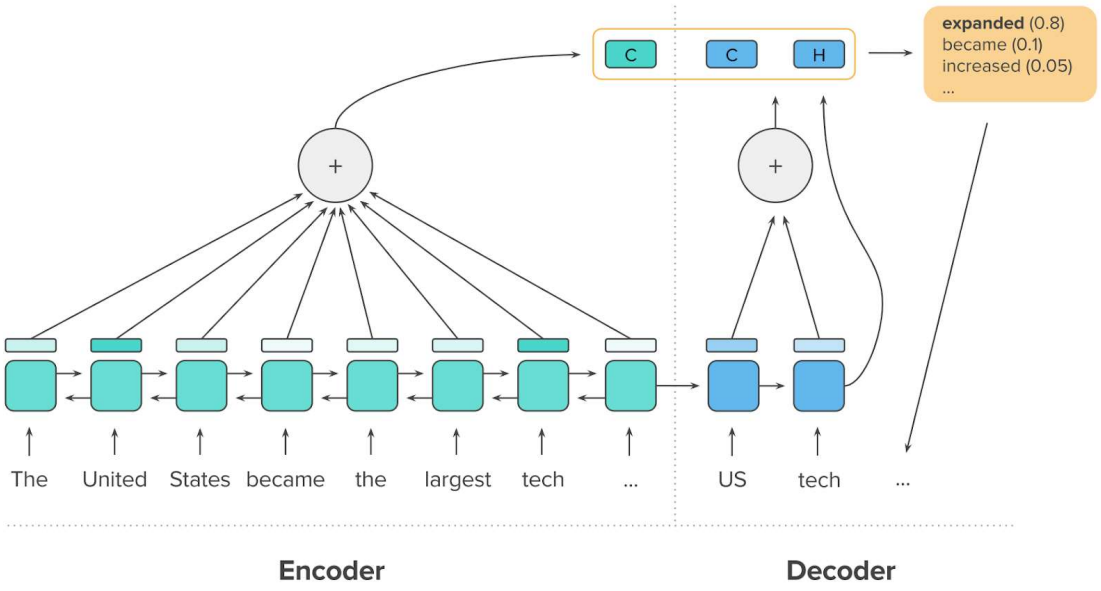


Figura 24: Atenções e vetores de contexto do modelo do artigo 5.

Fonte: Paulus et al. (2017).

O modelo proposto no artigo 5 utiliza atenção tanto no encoder quanto no decoder. A atenção no decoder, denominada intra-decoder, compara o estado oculto da etapa atual do decoder com os estados ocultos das etapas anteriores do decoder. Assim como tem duas atenções, o modelo do artigo 5 também tem dois vetores de contexto, que são concatenados (figura 24) com o estado oculto do decoder e utilizados para gerar a distribuição de vocabulário.

2.5.2 Matriz de representação

Os modelos estudados recebem a sentença de entrada em texto plano e transformam as palavras em vetores de representação. Esses vetores formam uma matriz de representação, no qual cada linha representa uma palavra. Geralmente, o encoder e o decoder contém uma matriz de representação cada. Entretanto, o modelo do artigo 5 compartilha uma matriz de representação única para os dois, pois a perda de desempenho é baixa comparada à diminuição de parâmetros do modelo.

2.5.3 Nova função objetivo

O artigo 5 também propõe uma nova função objetivo, que combina aprendizado supervisionado, utilizado nos artigos anteriores, e aprendizado por reforço, no qual o modelo é otimizado para uma métrica específica. O aprendizado supervisionado gera boas generalizações e, consequentemente, bons sumários. Entretanto, a melhora na qualidade dos sumários não é necessariamente acompanhada de melhora nas métricas. De forma semelhante, melhoria nas métrica não necessariamente acarreta em melhorias na qualidade dos sumários.

Buscando melhorar as notas nas métricas mantendo a qualidade dos sumários gerados pelo aprendizado supervisionado, o artigo 5 propõe uma função objetivo que combina aprendizado supervisionado com aprendizado por reforço, no qual o modelo é otimizado para uma métrica específica utilizando a mesma como função de recompensa durante o treinamento. No caso do artigo 5, a métrica escolhida é a ROUGE.

3. DESENVOLVIMENTO DO TRABALHO PROPOSTO

Nesta seção são descritos os métodos e técnicas utilizadas para a pesquisa e análise dos artigos sobre sumarização abstrativa, além dos resultados dos experimentos e análises. O conteúdo é apresentado em três subseções: pesquisa, experimentos e resultados.

3.1 METODOLOGIA

Para início do trabalho, foi decidido selecionar artigos com data de publicação entre 2015 e 2017, visando que trabalhos futuros abordassem o outro triênio (2017 até 2020). Foram selecionados cinco artigos, que utilizavam a abordagem abstrativa, avançavam no estado da arte e utilizavam *deep learning*. Os cinco artigos selecionados foram analisados, e os dois mais recentes foram escolhidos para um estudo detalhado e aprofundado, com o objetivo de entender as técnicas utilizadas no processo de geração de sumários dos modelos apresentados.

Para a etapa de experimento, foram procuradas implementações dos dois modelos estudados, sendo inicialmente treinados em computadores domésticos. Entretanto, devido ao longo tempo de treinamento, ficou perceptível a necessidade de utilizar equipamentos mais robustos. Para verificar as implementações encontradas, foram executados treinamentos com uma quantidade pequena de dados.

Após a confirmação do funcionamento das implementações, por meio de pesquisas, foi encontrada a plataforma Google Colab, que permitia a execução dos treinamentos por um período de doze horas contínuas. Como os modelos permitiam pausa dos treinamentos, foi possível utilizar a plataforma para a execução dos modelos.

Finalmente, com o término dos treinamentos, foram feitos testes com a mesma métrica utilizada nos artigos (ROUGE), com os resultados sendo comparados com os dados expostos nas publicações. A métrica ROUGE compara subsequências comuns entre o sumário de referência e o sumário gerado pelo modelo, assim como explicado na seção 2.2.1. Em seguida, foram escolhidos três sumários de cada um dos dois modelos treinados para análise da efetividade do resumo gerado, verificando o quão bem ele representa o texto original.

3.2 PESQUISA

Nesta subseção será tratada a etapa de pesquisa do trabalho, apresentando o método utilizado para a pesquisa.

3.2.1 Critérios

O tipo de pesquisa escolhido para o trabalho proposto foi a pesquisa exploratória, tendo início na elaboração de critérios que serviram para a orientação das buscas. Foram definidos três critérios para a seleção de artigos:

- data de publicação a partir do ano de 2015;
- utilização de *deep learning* na solução proposta;
- contribuição para o estado da arte de sumarização abstrativa no momento da publicação.

3.2.2 Triagem

Foram selecionados cinco artigos, conforme os critérios de seleção estabelecidos, cujas propostas foram descritas em seções do capítulo 2:

- artigo 1: A Neural Attention Model for Sentence Summarization (Rush, Chopra and Weston, 2015), descrito na seção 2.3, que utiliza um modelo de atenção local na geração das palavras do sumário;
- artigo 2: Abstractive Sentence Summarization with Attentive Recurrent Neural Networks (Chopra, Auli and Rush, 2016), descrito na seção 2.4, no qual é proposto um novo modelo de rede neural recorrente convolucional com atenção condicional;
- artigo 3: Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond (Nallapati et al., 2016), descrito na seção 2.5, no qual são propostos modelos que buscam reduzir os problemas encontrados em trabalhos anteriores, além de uma nova base de treinamento para modelos de sumarização com múltiplas sentenças;

- artigo 4: Get to the Point: Summarization with Pointer-generator Networks (See, Liu and Manning, 2017), descrito na seção 2.6, no qual é proposto um modelo que gera sumários com múltiplas sentenças, além de utilizar ponteiros para tratar palavras desconhecidas;
- artigo 5: A Deep Reinforced Model for Abstractive Summarization (Paulus, Xiong and Socher, 2017), descrito na seção 2.7, no qual é proposto um modelo que utiliza um novo método de treinamento que combina geração supervisionada de palavras e aprendizado por reforço.

3.3 EXPERIMENTOS

Para limitar o escopo dos experimentos, de forma a viabilizar o entendimento detalhado e a execução do código, o tempo de execução de aplicações complexas envolvendo *deep learning*, foram realizados treinamentos, testes e análises dos dois artigos estudados com data de publicação mais recentes (publicados em 2017): artigos 4 e 5.

A implementação do artigo 4 foi disponibilizada pelo autor[2]. Para o artigo 5 não foi disponibilizada nenhuma implementação oficial pelo autor. Para contornar essa questão, foi utilizada uma implementação disponibilizada em outra publicação, Keneshloo et al. (2019)[3], na qual foram analisados alguns modelos de sumarização abstrativa, englobando a proposta do artigo 5.

Durante a etapa de início de treinamento, os pesos da rede são preenchidos com valores aleatórios. Por razão dessa randomicidade, não é garantido que o modelo sempre chegará exatamente nos mesmos pesos finais, pois o modelo treinado não encontrará a solução ótima, apenas uma aproximação. Os modelos recebem os dados de treinamento várias vezes, até o erro do modelo estabilizar, finalizando o treinamento. Os dados de teste geralmente são compostos por uma parte dos dados de treinamento, tendo tamanhos menores. Contudo, diferente dos dados de treinamento, o modelo recebe os dados de teste apenas uma vez, já que os pesos foram ajustados durante o treinamento e

[2] https://github.com/abisee/pointer-generator
[3] https://github.com/yaserkl/RLSeq2Seq

não são mais atualizados. Os sumários de saída do modelo são pontuados a partir da métrica ROUGE, explicada na seção 3.4.

As limitações impostas pela plataforma utilizada para execução dos treinamentos e testes não impediu o uso da mesma, pois as duas implementações utilizadas permitiam a pausa do treinamento, com os pesos da rede sendo salvos no disco. Assim, quando o treinamento era executado novamente, os pesos salvos eram utilizados, não necessitando iniciar o treinamento desde o início.

Nesta seção são comentados os hiperparâmetros e os dados de treinamento dos dois artigos, com o treinamento em si sendo comentado nas subseções 3.2.2 e 3.2.5, nas quais são comentados os tempos e as condições de treinamento. Os testes e análise dos resultados são comentados na próxima seção.

3.3.1 Artigo 4 - Hiperparâmetros e dados de treinamento

A função de otimização utilizada foi Adagrad (Duchi et al., 2011) com taxa de aprendizado de 0.15, com o tamanho de *batch* sendo 16. O modelo contém 256 estados ocultos, representação de palavra com 128 de dimensão e com o vocabulário fixo contendo 50 mil *tokens* tanto para a entrada quanto para a saída.

Durante o treinamento e teste, os artigos de entrada foram truncados em 400 *tokens* e os sumários tiveram os tamanhos limitados em 100 *tokens* durante o treinamento e 120 *tokens* durante os testes, o que, segundo o autor, acelera o treinamento além de melhorar o desempenho do modelo.

Durante a etapa de testes, os sumários são gerados utilizando busca por feixe com comprimento 4. Seguindo indicações dos autores, os hiperparâmetros que definem os valores máximos de etapas do *encoder* e *decoder* foram iniciados com valores menores, 100 e 25 respectivamente, com o objetivo de obter iterações mais rápidas no início do treinamento, sendo aumentados até chegar em 400 para *encoder* e 100 para *decoder*.

O conjunto de dados utilizado para treinamento e testes foi o CNN/Daily Mail (Herman, et al., 2015; Nallapati et al., 2016), que contém artigos de notícias com 781 *tokens* em média, com sumários de múltiplas sentenças contendo 56 *tokens* em média.

3.3.2 Artigo 4 - Treinamento

Para a etapa de treinamento do modelo, foi utilizada a plataforma *Google Colab*[4] , que disponibiliza processamento na nuvem utilizando GPU Tesla K80 de forma gratuita, por tempo limitado durante cada execução (12 horas contínuas). Devido ao fato de que o modelo disponibilizado pelo autor suporta pausa do treinamento, a limitação de tempo de execução não inviabilizou o uso da plataforma.

O tempo de treinamento do modelo, informado no artigo 4, foi de 3 dias e 4 horas utilizando uma GPU Tesla K40, com aproximadamente 230 mil iterações, o que representa aproximadamente 12,8 epochs[5]. Na execução do experimento, o treinamento do modelo levou aproximadamente 3 dias para o mesmo número de iterações reportadas no artigo 4 resultando, portanto em tempos próximos de treinamento.

3.3.3 Artigo 5 - Hiperparâmetros e dados de treinamento

Para treinamento do modelo sem aprendizado por reforço, Paulus et al. (2017) utiliza o algoritmo de aprendizado por reforço com uma diferença: 25% de chance de utilizar o *token* gerado pela iteração anterior do decoder como entrada, ao invés do *token* do sumário de referência, o que reduz o viés de exposição segundo Venkatraman et al. (2015). Para a função de perda que combina ML+RL, foi utilizado o fator de escala $\gamma = 0.9984$.

São utilizadas duas LSTMs com 200 estados ocultos para o *encoder* bidirecional e uma LSTM com 400 estados ocultos para o *decoder*. O tamanho do vocabulário utilizado pelo autor foi 150.000 *tokens* para entrada, enquanto o vocabulário de saída é limitado em 50.000 *tokens* que aparecem com maior frequência nos dados de entrada. A função de ativação utilizada foi Adam (Kingman & Ba, 2014), com o tamanho de *batch* igual a 50 e taxa de aprendizado de 0.001 para ML e 0.0001 para RL. Na etapa de treinamento, a busca por feixe utiliza comprimento 5 para gerar as predições finais.

No artigo, o autor expõe resultados para dois conjuntos de dados: CNN/Daily Mail (Herman, et al., 2015; Nallapati et al., 2016) e New York Times (Sandhaus, 2008).

[4] https://colab.research.google.com/

[5] Um *epoch* equivale a passar todos os dados de treinamento pelo modelo, enquanto uma iteração consiste em passar um *batch* pelo modelo.

Para efeito de comparação entre os resultados do artigo4 e 5, foi escolhido o conjunto CNN/Daily Mail nos experimentos.

3.3.4 Artigo 5 - Treinamento

Assim como no artigo anterior, toda a etapa de treinamento foi realizada na plataforma *Google Colab*. Como o modelo disponibilizado por Keneshloo et al. (2019) permitia a pausa do treinamento, a limitação de tempo já citada não proibiu o uso da plataforma. Todavia, a memória disponível no *Google Colab* (12 GB) não permitiu que fossem utilizados os mesmos tamanhos de *batch* e vocabulário. O valor inferior de *batch size* apenas interfere no tempo para treinamento, mas o tamanho do vocabulário pode interferir nos resultados do modelo. Na execução do experimento, o treinamento do modelo levou aproximadamente 7 dias e 4 horas, com aproximadamente 412 mil iterações executadas, seguindo o processo utilizado pelo autor da implementação. Paulus et al. (2017) não disponibilizou o tempo de treinamento e a quantidade de iterações em seu trabalho.

3.4 RESULTADOS DOS TESTES

Para a etapa de avaliação, seguindo o artigo 4, foi utilizada a métrica ROUGE (Recall-Oriented Understudy for Gisting Evaluation**)** (Lin, 2004), baseado em BLEU (Papineni et al., 2001), outra métrica muito utilizada para avaliação de traduções de máquinas automáticas. A métrica ROUGE opera comparando um sumário criado automaticamente com um sumário de referência, produzido por humanos. Para isso, a métrica leva em consideração ocorrências de sequências de palavras nos dois sumários, sendo disponibilizadas cinco medidas:

- **ROUGE-1:** estima a média de vezes que cada palavra é encontrada em cada um dos sumários. As sequências contendo uma só palavra também são denominadas uni-gram;
- **ROUGE-2:** parecido com o ROUGE-1, mas calcula os bi-grams, sequências contendo duas palavras consecutivas, nos dois sumários;

- **ROUGE-3:** seguindo as outras medidas, calcula a média de tri-grams (sequências contendo três palavras consecutivas) nos dois sumários. Não é muito utilizado nas avaliações, já que sequências acima de duas palavras são difíceis de ocorrer);
- **ROUGE-4:** calcula a média de tetra-grams (sequência de quatro palavras consecutivas) nos dois sumários. Assim como o ROUGE-3, não é muito utilizado;
- **ROUGE-L:** calcula a mesma avaliação dos n-grams, mas com a maior subsequência comum (Longest Common Subsequence, LCS) entre os dois sumários. Caso uma subsequência englobe outra subsequência anterior, apenas a última é considerada.

O tempo para gerar os sumários dos dados de teste foi de aproximadamente 7 horas. As pontuações são apresentadas no quadro abaixo:

MODELO	ROUGE-1	ROUGE-2	ROUGE-L
Artigo	39.53	17.28	36.38
Experimento	38.53	16.75	35.48

Quadro 1 - Pontuação do modelo treinado e See et al. (2017).

Fonte: Elaborado pelo autor.

Analisando os resultados, a pontuação do modelo que foi treinado para o experimento é levemente menor que a apresentada no artigo. Isso era esperado, já que o próprio autor explica, na página na qual disponibilizou, que os resultados obtidos com o modelo do artigo não foram mantidos ao treinar outro modelo com as mesmas configurações. Sabendo disso, era esperado que o treinamento não resultasse nas mesmas pontuações apresentadas pelo autor.

Entretanto, mesmo com essa pequena diferença, o modelo treinado para o experimento ainda supera os modelos de sumarização abstrativa com aprendizado profundo que representavam o estado da arte na época (Nallapati et al., 2016), o que acompanha os resultados demonstrados por See et al. (2017).

Para o artigo 5, a métrica ROUGE foi utilizada para avaliação. As pontuações são apresentadas no quadro abaixo, nas quais todos os modelos abaixo utilizam intra-atenção:

MODELO	ROUGE-1	ROUGE-2	ROUGE-L
Artigo (RL)	41.16	15.75	39.08
Artigo (ML+RL)	39.87	15.82	36.90
Experimento (RL)	40.52	15.25	38.79
Experimento (ML+RL)	39.10	15.20	36.27

Quadro 2 - Pontuação do modelo treinado com Paulus et al. (2017).

Fonte: Elaborado pelo autor.

As pontuações dos modelos que foram treinados para o experimento diferem das apresentadas no artigo, sendo levemente menores no geral. As diferenças nos valores de alguns parâmetros de execução (tamanhos do *batch* e vocabulário) podem ser responsáveis pelas diferenças nos resultados.

Contudo, os modelos treinados ainda superam os modelos de sumarização abstrativa com aprendizado profundo que representavam o estado da arte na época (Nallapati et al., 2016), o que acompanha os resultados demonstrados por Paulus et al. (2017). Adicionalmente, os resultados do modelo do artigo 5 são próximos do modelo do artigo 4, sendo que a diferença não é significante o suficiente para declarar um dos dois como o melhor.

3.5 ANÁLISE DE RESUMOS GERADOS

A seguir são ilustrados alguns exemplos de sumários gerados pelos modelos treinados dos artigos 4 e 5, seguidos de análise de qualidade dos mesmos:

A1 - Texto original:

washington -lrb- cnn -rrb- israeli prime minister benjamin netanyahu criticized the deal six world

powers struck to thwart iran 's nuclear ambitions , saying he sees better options than `` this bad deal or war . '' `` i think there 's a third alternative , and that is standing firm , ratcheting up the pressure until you get a better deal , '' netanyahu told cnn 's jim acosta sunday on `` state of the union . '' his comments come as democrats and republicans spar over the framework announced last week to lift western sanctions on iran in exchange for the country dropping from 19,000 to 5,060 active centrifuges , limiting its highly enriched uranium , and increasing inspections . president barack obama endorsed the deal , saying it was better than the alternatives . but gop contenders for the party 's 2016 presidential nomination lambasted it , saying it gave iran too much flexibility . on sunday , the sparring continued . one senate democrat said netanyahu needs to `` contain himself . '' and a top republican said almost any of obama 's successors as president `` could do better . '' netanyahu 's most recent argument against the iran nuclear deal was similar to the one he 'd made in a march trip to washington , when he addressed a joint session of congress -- fueling a republican push to have the deal sent to congress before it 's implemented . `` it does not roll back iran 's nuclear program . it keeps a vast nuclear infrastructure in place . not a single centrifuge is destroyed . not a single nuclear facility is shut down , including the underground facilities that they built illicitly . thousands of centrifuges will keep spinning , enriching uranium , '' netanyahu said sunday . `` that 's a very bad deal . '' netanyahu said iran is a country of `` congenital cheating '' and that it ca n't be trusted to abide by the terms of the deal , which lasts 10 years with some provisions extending well beyond that . he said his opposition has little to do with his frosty relationship with obama . `` i think that we can have a legitimate difference of opinion on this , because i think iran has shown to be completely distrustful , '' netanyahu said . democratic sen. dianne feinstein of california , meanwhile , said she wishes netanyahu `` would contain himself . '' the top-ranking democrat on the senate intelligence committee said negotiators working on the deal -- from iran and the united states , as well as russia , china , britain , france and germany -- are `` on the cusp of something that can be workable . '' `` it 's a framework . it has to be wrapped into a final agreement . there still can be some changes , '' feinstein said . `` but i do n't think it 's helpful for israel to come out and oppose this one opportunity to change a major dynamic -- which is downhill , a downhill dynamic in this part of the world . '' energy secretary ernest moniz defended the deal in an appearance on cbs ' `` face the nation '' on sunday , saying it would extend from two months to one year the `` breakout '' time period -- the length of time it would take iran to build a nuclear bomb . he said it also allows for the `` almost instantaneous recognition of any attempt to evade the deal . '' `` we have blocked all of these pathways to a bomb , '' he said . sen. lindsey graham , r-south carolina , said on `` face the nation '' that the best option for the united states is to keep current sanctions in place for two more years and then have a `` new crack at it with a new president that does n't have the baggage of obama . '' and he said the alternatives to obama on both sides -- with the exception of sen. rand paul of kentucky , who 's called for a less active u.s. role overseas -- would likely strike a better deal . `` hillary clinton would do better . i think everybody on our side , except , maybe , rand paul , could do better , '' graham said .

A2 - Sumário de referência:

netanyahu says third option is `` standing firm '' to get a better deal .
political sparring continues in u.s. over the deal with iran .

A3 - Sumário gerado pelo modelo do artigo 4:

israeli prime minister benjamin netanyahu criticized the deal six world powers .
his comments come as democrats and republicans spar over the framework .
but gop contenders for the party 's 2016 presidential nomination lambasted it .
netanyahu 's most recent argument against the iran nuclear deal was similar to the one he 'd made .

A4 - Sumário gerado pelo modelo do artigo 5:

israeli prime minister benjamin netanyahu criticized the deal six world powers .

president obama endorsed the deal , saying it was better than the alternatives .

netanyahu says there 's a third alternative .

B1 - Texto original:

a 19-year-old man was charged wednesday with terror offenses after he was arrested as he returned to britain from turkey , london 's metropolitan police said . yahya rashid , a uk national from northwest london , was detained at luton airport on tuesday after he arrived on a flight from istanbul , police said . he 's been charged with engaging in conduct in preparation of acts of terrorism , and with engaging in conduct with the intention of assisting others to commit acts of terrorism . both charges relate to the period between november 1 and march 31 . rashid is due to appear in westminster magistrates ' court on wednesday , police said . cnn 's lindsay isaac contributed to this report .

B2 - Sumário de referência:

london 's metropolitan police say the man was arrested at luton airport after landing on a flight from istanbul .
he 's been charged with terror offenses allegedly committed since the start of november .

B3 - Sumário gerado pelo modelo do artigo 4:

yahya rashid , a uk national from northwest london , was detained at luton airport .

he 's been charged with engaging in conduct in preparation of acts of terrorism .

both charges relate to the period between november 1 and march 31 .

B4 - Sumário gerado pelo modelo do artigo 5:

yahya rashid was detained at luton airport on tuesday .

19-year-old man charged with terror offenses .
he is been charged with engaging in conduct in preparation of acts of terrorism .
rashid is due to appear in westminster magistrates ' court on wednesday .

C1 - Texto original:

the fbi charged a philadelphia woman on thursday with trying to travel overseas to fight for isis . she 's one of three women arrested this week on terror charges . two new york women were also taken into custody . an fbi complaint cites numerous social media messages dating back to august 2013 that were sent by keonna thomas , 30 , also known as `` young lioness " and `` fatayat al khilafah . " one twitter message said , `` if we truly knew the realities ... we all would be rushing to join our brothers in the front lines pray allah accept us as shuhada -lsb- martyrs -rsb- . " another said , `` when you 're a mujahid -lsb- violent jihadi fighter -rsb- your death becomes a wedding . " the fbi said thomas purchased an electronic visa to turkey on march 23 . turkey is known as the easiest place from which to enter syria and join isis . an isis manual advises recruits to buy round-trip tickets to vacation spots such as spain and then purchase tickets for their real destination once they arrive overseas , the fbi said . on march 26 , thomas purchased a ticket to barcelona , with a march 29 departure and an april 15 return to the united states , the complaint said . it 's not clear when or where she was arrested . she was charged with knowingly attempting to provide material support and resources to a designated foreign terrorist organization . she could be sentenced to 15 years in prison . on thursday , noelle velentzas , 28 , and her former roommate , asia siddiqui , 31 , were arrested in new york and accused of planning to build an explosive device for attacks in the united states , federal prosecutors said . in the past 18 months , the justice department 's national security division has prosecuted or is prosecuting more than 30 cases of people attempting to travel abroad to join or provide support to terrorist groups . of those cases , 18 allegedly involve support to isis . `` the terrorist threat is more decentralized , more diffuse , more complicated , " homeland security secretary jeh johnson told reporters thursday . `` it involves the potential lone wolf actor , it involves the effective use of social media , the internet . "

C2 - Sumário de referência:

the fbi cites social media messages sent by keonna thomas , 30 . she 's accused of trying to travel overseas to join isis . thomas is one of three women facing federal terror charges this week .

C3 - Sumário gerado pelo modelo do artigo 4:

the fbi charged a philadelphia woman on thursday with trying to travel overseas to fight for isis .
she 's one of three women arrested this week on terror charges .
two new york women were also taken into custody .

C4 - Sumário gerado pelo modelo do artigo 5:

the fbi charged a woman on thursday with trying to travel overseas to fight for isis .

she 's one of three women arrested this week on terror charges .

fbi complaint cites social media messages dating back to august 2013 .

A sumarização abstrativa utilizando deep learning ainda está em seus passos iniciais. Embora os resultados atuais sejam positivos, há um caminho longo para ser percorrido até ser possível gerar sumários com o mesmo nível dos sumários feitos por humanos. Melhoria nas métricas não necessariamente garante melhor qualidade e legibilidade nos sumários.

No quadro acima, são apresentados alguns exemplos de sumários gerados pelos modelos do artigo 4 e artigo 5. No exemplo A, a principal informação repassada pelo texto fonte A1 é que o primeiro-ministro de Israel, Benjamin Netanyahu, criticou o acordo proposto com o Irã, afirmando que acordos melhores são possíveis. O texto expõe algumas falas de pessoas envolvidas na política americana, mas que não são necessárias para o entendimento do ocorrido. Embora o texto fonte seja bem maior que o sumário de referência A2, a informação principal continua sendo repassada.

O sumário A3, gerado pelo modelo do artigo 4, não consegue manter a mesma qualidade do sumário de referência, sendo incorreto na primeira sentença, onde o nome do acordo é informado errado, como se fosse denominado "*six world powers"*, incompleta na terceira sentença, onde a palavra "*party*" está fora de contexto, e incompleto na última sentença, que termina de forma abrupta, omitindo o resto da sentença. Neste primeiro exemplo, o sumário gerado não consegue transmitir as informações principais do texto. Adicionalmente, todas as sentenças que compõem o sumário foram retiradas do texto fonte, sem adição de palavras ou mudanças na ordem das palavras e estrutura da frase.

O sumário A4, gerado pelo modelo do artigo 5, também comete o mesmo erro do sumário A3 na primeira sentença: trata o acordo como se fosse denominado "*six world power*s". Entretanto, a segunda segunda sentença é correta, embora seja copiada inteiramente do texto fonte. A terceira sentença é correta, além de ser gerada pelo modelo, sem copiar inteiramente da entrada, mesmo terminando de modo abrupto. Em comparação ao sumário A3, gerado pelo artigo 4, é um sumário mais correto, mas não atinge a qualidade do sumário de referência A2.

No exemplo B, a principal informação repassada pelo texto fonte B1 é que um homem foi preso no aeroporto de Londres acusado de crimes de terrorismo. O texto complementa a informação principal informando datas e relatos da polícia. O sumário de referência B2 consegue passar a informação contida no texto fonte, omitindo os fatos que não são importantes para o entendimento da notícia.

O sumário B3, gerado pelo modelo do artigo 4, acerta nas duas primeiras sentenças, com informações corretas, embora a segunda sentença seja incompleta, pois não comenta sobre a segunda acusação sobre ajudar pessoas que tem a intenção de cometer atos de terrorismo. Contudo, na última sentença, acaba utilizando a palavra "*both*" sem ter citado a outra acusação na segunda sentença, o que pode ser visto como um erro. As três sentenças foram copiadas inteiramente da entrada, demonstrando que o uso do gerador de ponteiros pode acarretar em uma tendência extrativa além de nem

sempre garantir a reprodução correta dos fatos.

O sumário B4, gerado pelo modelo do artigo 5, também é correto nas três primeiras sentenças, sofrendo do mesmo problema do sumário B3, no qual apenas uma das acusações é exposta. Entretanto, essa é a única falha que pode ser considerada relevante no sumário. Assim como no sumário B3, o modelo copiou sentenças inteiras do texto fonte, com a diferença que a saída não é composta unicamente de cópias. A primeira sentença é correta, sendo gerada pelo modelo, sem cópia da entrada. A segunda é correta, sendo copiada da entrada. A terceira é incompleta, já que omite a outra acusação do homem apreendido. A quarta é correta, sendo copiada inteiramente da entrada. Ademais, diferentemente do sumário B3, a ordem das sentenças é diferente do texto fonte B1, o que pode ser visto como influência do aprendizado por reforço, já que o mesmo não é limitado a comparação por palavras durante o treinamento, mas por sentenças.

No exemplo C, a principal informação repassada pelo texto fonte C1 é que uma mulher foi acusada de tentar viajar para apoiar o grupo terrorista Estado Islâmico (ISIS), sendo uma das três mulheres presas na mesma semana por crimes de terrorismo. O sumário de referência C2 consegue passar as informações mais relevantes do texto fonte, embora também contenha informações que poderiam ser omitidas, como o nome e a idade da mulher presa.

O sumário C3, gerado pelo modelo do artigo 4, é correto nas suas três sentenças. Entretanto, o sumário é totalmente composto por sentenças completas copiadas do texto fonte, sendo exatamente as três primeiras sentenças que compõem o texto de entrada. Essa abordagem é a mesma do modelo extrativo lead-3, usado no artigo para comparação. Segundo a autora, o gerador de ponteiro induz o modelo a uma tendência extrativa, o que explica a geração de sumários apenas copiando sentenças completas da entrada.

O sumário C4, gerado pelo modelo do artigo 5, também é composto, em partes, por sentenças completas tiradas do texto fonte. As duas primeiras sentenças do sumário são as duas primeiras sentenças do texto fonte. Entretanto, a última sentença, embora composta por partes de uma sentença da entrada, não é copiada completamente, omitindo informações não necessárias para o entendimento da sentença e transmitindo a informação principal repassada pela sentença no texto original, sobre as mensagens de redes sociais da acusada.

A1 (texto original)

- primeiro-ministro de Israel, Benjamin Netanyahu, criticou o acordo proposto com o Irã, afirmando que acordos melhores são possíveis
- texto expõe algumas falas de pessoas envolvidas na política americana, mas que não são necessárias para o entendimento do ocorrido.

A2 (sumário de referência)

- bastante compacto, mas repassa a informação essencial do texto

A3 (sumário do artigo 4)

- características gerais
    - todas as sentenças que compõem o sumário foram retiradas do texto fonte
        - sem adição de palavras ou mudanças na ordem das palavras e estrutura da frase
    - não consegue transmitir as informações principais do texto
- análise por sentença
    - primeira sentença: incorreta (o acordo ocorreu entre as 6 potências mundiais, mas esse não é o título do acordo)
    - segunda sentença : correta, mas copia o texto original
    - terceira sentença : incompleta (não cita o nome do partido)
    - última sentença : incompleta (omite quando o argumento anterior for feito)

A4 (sumário do artigo 5)

- características gerais
    - em geral, não replica o texto original
    - caracteriza melhor as principais informações do texto
- análise por sentença
    - primeira sentença : mesmo problema da primeira sentença de A3
    - segunda sentença : correta, embora idêntica ao texto original
    - terceira sentença : correta, sem replicar o texto original; mas incompleta (cita uma terceira alternativa sem qualificá-la)

Comparação entre A3 e A4

- A4 é mais abstrativa, pois compõem várias sentenças; enquanto todas a sentenças geradas em A3 são retiradas do texto original
- ambas têm sentenças incompletas
- ambas não atingem o nível de abstração do sumário de referência

B1 (texto original)

- homem foi preso no aeroporto de Londres acusado de terrorismo
- texto expõe datas e relatos da polícia sobre o ocorrido

B2 (sumário de referência)

- idem A2

B3 (sumário do artigo 4)

- Características gerais
    - assim como A3, todas as sentenças que compõem o sumário foram retiradas do texto fonte

    - não consegue transmitir as informações principais do texto
- Análise por sentença
    - primeira sentença: correta, mas copiada do texto original
    - segunda sentença: incompleta, pois omite uma das acusações
    - terceira sentença : correta, mas também é copiada do texto original

B4 (sumário do artigo 5)

- Características gerais
    - copia sentenças inteiras do texto original, mas a saída não é composta só de cópias
    - ordem das sentenças difere do texto
- Análise por sentença
    - primeira sentença: correta, gerada pelo modelo
    - segunda sentença: correta, mas copiada da entrada
    - terceira sentença: incompleta, já que omite uma das acusações
    - última sentença: correta, porém cópia do texto original

Comparação entre B3 e B4

- B4 é mais abstrativa, também copiando da entrada mas mudando a ordem
- assim como em A3 e A4, ambas têm sentenças incompletas e não atingem o nível de abstração do sumário de referência

C1 (texto original)

- mulher foi acusada de tentar viajar para apoiar grupo terrorista
- texto expõe detalhes do caso e comenta sobre outros casos semelhantes em datas próximas

C2 (sumário de referência)

- idem A2 e B2, mas contém informações que poderiam ser omitidas sem muita perda de informação

C3 (sumário do artigo 4)

- Características gerais
    - assim como A3 e B3, todas as sentenças que compõem o sumário foram retiradas do texto fonte
    - formado pelas três primeiras sentenças da entrada
- Análise por sentença
    - primeira sentença: correta, mas copiada do texto original
    - segunda sentença: idem
    - terceira sentença : idem

C4 (sumário do artigo 5)

- Características gerais
    - assim como B4, copia sentenças da entrada mas a saída não é composta só de cópias
    - ordem das sentenças difere do texto
- Análise por sentença
    - primeira sentença: correta, gerada pelo modelo
    - segunda sentença: correta, mas copiada da entrada
    - terceira sentença: correta, omite palavras do texto original sem deixar de transmitir a informação principal

Comparação entre C3 e C4

- assim como A4 e B4, C4 é mais abstrativa que C3. Porém os dois sumários ainda são compostos em grande parte por cópias da entrada.

4. CONSIDERAÇÕES FINAIS

Nesta seção são apresentadas as conclusões, as dificuldades encontradas e uma sugestão para trabalhos futuros.

4.1 CONCLUSÕES

Foram estudadas cinco propostas de aplicação de *deep learning* à sumarização abstrativa. Particularmente, as duas propostas mais recentes foram estudadas com muito mais detalhes, e pesquisadas implementações que foram utilizadas para treinamento e teste e análise comparativa dos resultados. Os resultados alcançados foram próximos aos resultados relatados nos artigos, com as diferenças sendo explicadas por uma questão de implementação, no artigo 4, e valores de parâmetros diferentes, no artigo 5, devido à condições limitantes durante a execução do treinamento, como a memória e processamento disponíveis na plataforma Google Colab.

Em relação a análise dos sumários gerados, o modelo treinado do artigo 4 tem uma grande tendência extrativa, copiando totalmente ou a maior parte da entrada. Essa tendência é destacada pelos autores do artigo 4, sendo a geração de ponteiros indicada como causa. Já o modelo do artigo 5 melhora em algumas questões do modelo 4, com melhores resultados na métrica ROUGE, copiando menos da entrada, embora ainda ocorra a utilização de sentenças inteiras da entrada na saída gerada. Contudo, o nível de qualidade dos sumários gerados pelos dois modelos é próximo, não sendo perceptível um salto de qualidade entre as saídas geradas.

Com o estudo finalizado, ficou perceptível que a área de sumarização abstrativa, embora com grandes avanços nos últimos anos, ainda tem uma grande jornada pela frente, com problemas recorrentes que ainda não foram solucionados, como as poucas sentenças dos sumários gerados e a repetição de frases.

4.2 DIFICULDADES ENCONTRADAS

Durante o desenvolvimento do trabalho, as principais dificuldades foram relacionadas à obtenção das implementações e execução das mesmas. Entre os artigos selecionados para análise, apenas dois tinham implementações oficiais disponibilizadas

pelos autores. Além disso, a dificuldade de entender os processos associados a cada proposta, agravada pelo tempo gasto para preparação para execução e para o treinamento inviabilizaram a execução dos cinco modelos propostos. Desse modo, decidiu-se por executar apenas os dois trabalhos mais recentes entre os selecionados.

4.3 TRABALHOS FUTUROS

Para trabalhos futuros, recomenda-se o estudo de publicações do próximo triênio, 2018 até 2020, além da utilização de um conjunto maior de exemplos para a análise, pois não foi possível analisar uma grande quantidade de exemplos, dado que é um processo manual e custoso.

REFERÊNCIAS

MANI, Inderjeet et al. The TIPSTER SUMMAC text summarization evaluation. **Ninth Conference of the European Chapter of the Association for Computational Linguistics**. 1999.

HILBERT, Martin. How much information is there in the “information society”?. **Significance**, v. 9, n. 4, p. 8-12, 2012.

TAS, Oguzhan; KIYANI, Farzad. A SURVEY AUTOMATIC TEXT SUMMARIZATION. **PressAcademia Procedia**, v. 5, n. 1, p. 205-213, 2007.

MORATANCH, N.; CHITRAKALA, S. A survey on abstractive text summarization. In: **2016 International Conference on Circuit, power and computing technologies (ICCPCT)**. IEEE, 2016. p. 1-7.

SUTSKEVER, Ilya. Training recurrent neural networks. Toronto, Canada: University of Toronto, 2013.

OLAH, Christopher. Understanding LSTM Networks. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> Acesso em 29 de out. de 2019.

Geeky is Awesome. Using beam search to generate the most probable sentence. Disponível em: <https://geekyisawesome.blogspot.ie/2016/10/using-beam-search-to-generate-most.html > Acesso em 24 de ago. de 2020.

RUSH, Alexander M. et al. A neural attention model for sentence summarization. In: **ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing**. 2017.

CHOPRA, Sumit; AULI, Michael; RUSH, Alexander M. Abstractive sentence summarization with attentive recurrent neural networks. In: **Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. 2016. p. 93-98.

NALLAPATI, Ramesh et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. **arXiv preprint arXiv:1602.06023**, 2016.

SEE, Abigail; LIU, Peter J.; MANNING, Christopher D. Get to the point: Summarization with pointer-generator networks. **arXiv preprint arXiv:1704.04368**, 2017.

BARBOSA, Anderson Henrique; FREITAS, Marcílio Sousa da Rocha; NEVES, Francisco de Assis das. Confiabilidade estrutural utilizando o método de Monte Carlo e redes neurais. **REM: Revista Escola de Minas**, v. 58, n. 3, p. 247-255, 2005.

PAULUS, Romain; XIONG, Caiming; SOCHER, Richard. A deep reinforced model for abstractive summarization. **arXiv preprint arXiv:1705.04304**, 2017.

KENESHLOO, Yaser et al. Deep reinforcement learning for sequence-to-sequence models. **IEEE Transactions on Neural Networks and Learning Systems**, 2019.

KOSTADINOV, Simeon. Understanding Encoder-Decoder Sequence to Sequence Model. Disponível em: <https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346> Acesso em: 29 de out. de 2019.

KANN, Katharina; COTTERELL, Ryan; SCHÜTZE, Hinrich. Neural morphological analysis: Encoding-decoding canonical segments. In: **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing**. 2016. p. 961-967.

BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

LIN, Chin-Yew; ROUGE: A Package for Automatic Evaluation of Summaries. In: **Text Summarization Branches Out.** jul. 2004. p. 74-81.

CHUNG, Junyoung et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint arXiv:1412.3555**, 2014.

SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to sequence learning with neural networks. In: **Advances in neural information processing systems**. 2014. p. 3104-3112.

VENUGOPALAN, Subhashini et al. Sequence to sequence-video to text. In: **Proceedings of the IEEE international conference on computer vision**. 2015. p. 4534-4542.

PRABHAVALKAR, Rohit et al. A Comparison of Sequence-to-Sequence Models for Speech Recognition. In: **Interspeech**. 2017. p. 939-943.

SUNDERMEYER, Martin; SCHLÜTER, Ralf; NEY, Hermann. LSTM neural networks for language modeling. In: **Thirteenth annual conference of the international speech communication association**. 2012.

BRITZ, Denny et al. Massive exploration of neural machine translation architectures. **arXiv preprint arXiv:1703.03906**, 2017.

KAISER, Łukasz; SUTSKEVER, Ilya. Neural gpus learn algorithms. **arXiv preprint arXiv:1511.08228**, 2015.

CHO, Kyunghyun et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

LUONG, Minh-Thang; PHAM, Hieu; MANNING, Christopher D. Effective approaches to attention-based neural machine translation. **arXiv preprint arXiv:1508.04025**, 2015.