Sumarização Automática Abstrativa Aplicada a Textos Longos em Português

Gabriel Vasconcelos
Departamento de Ciência de Computação
Universidade de Brasília
Brasília, Dsitrito Federal
Email: gcvasconcelos98@gmail.com

***Abstract*—Neste projeto, é proposta a aplicação de um modelo *sequence-to-sequence* usando Redes Neurais Recorrentes (RNN) implementados em uma arquitetura *encoder-decoder* com o objetivo da resolução da tarefa de sumarização automática abstrativa, ou seja, a construção de um resumo que capture o sentido e contexto presente no texto original mas que não utiliza necessariamente as palavras deste. Os textos estão escritos na língua portuguesa e são transcrições de pronunciamentos realizados por senadores no Senado Federal do Brasil. A base têm a característica especial de possuir um extenso vocabulário e tamanho onde, todavia, cada discurso discorre sobre pouco mais de um assunto geral. Para lidar com esses desafios, na fase de treinamento do modelo foi utilizado no *encoder* um conjunto de modelos *Long Short-Term Memory* (LSTM) bi-direcional que é treinado com os textos dos discursos e tem como fim capturar seu contexto, tanto do passado como do futuro. Partes importantes deste contexto são selecionadas por um mecanismo de *attention* que é utilizado para inicializar o *decoder*, uma outra LSTM que é treinada com resumos gerados por humanos. Os resumos foram gerados automaticamente para 14.535 discursos e avaliados a partir das métricas ROUGE e BERTScore, que tiveram um resultado satisfatório quando comparados com a literatura.**

I. Introdução

Os crescentes avanços tecnológicos agora permitem que uma quantidade numerosa de dados, que antes eram guardados de forma física ou apenas na memória das pessoas, sejam guardados na forma digital em discos rígidos pessoais ou em grandes servidores espalhados ao redor do mundo. Destes dados, os mais fáceis de ser tanto produzidos como consumidos por pessoas comuns estão disponibilizados em forma de texto escrito, sejam em notícias, blogs ou postagens de redes sociais.

Entretanto, esse grande volume de informação cria uma situação similar a encontrar uma agulha no palheiro e muitas vezes existe a necessidade da seleção do que é ou não é relevante para ser consumido sobre determinado assunto ou objeto de pesquisa. E com a crescente inclusão digital da população mundial, esse problema torna-se cada vez mais um problema global.

A sumarização de textos consiste no processo de diminuir o tamanho de um texto mantendo seu significado principal. ou seja, seu contexto, bem como seus pontos principais abordados no documento original. A principal motivação neste processo é economizar tempo do leitor em atividades que necessitam de seleção de grandes corpos de texto, como na pesquisa acadêmica, em processos burocráticos do direito ou na própria indexação de documentos em geral.

Porém, a sumarização manual é muito custosa em termos do tempo gasto e da dificuldade do trabalho, que deve levar em conta uma série de considerações como tamanho, estilo de escrita e semântica do texto. Por isso a procura por uma forma de automatização desta tarefa é um tópico que vem sendo explorado desde o início deste século e mais recentemente facilitado pelos avanços nas áreas de processamento de linguagem natural e aprendizagem profunda [1].

Os métodos atuais de sumarização podem ser agrupados em duas categorias. A **extrativa** foca em escolher as sentenças mais importantes do texto deixando o seu conteúdo e ordem inalterada. Já na **abstrativa**, tenta-se entender a semântica do texto como um todo e a partir disto gerar um conjunto de novas sentenças que expressem seus significados-chave [2].

O objetivo deste projeto é gerar sumários, também chamados de resumos, dos textos de uma base de discursos de senadores realizados no Senado Federal do Brasil utilizando o método abstrativo. Esse resumo poderá ser usado para facilitar o acesso aos discursos realizados na instituição dentro dos campos de busca do site, fornecendo uma breve explicação sobre o que é falado no discurso completo. Os dados foram obtidos utilizando a API REST disponibilizada no *site* da instituição [3], onde foram coletados mais de 70.000 discursos e seus respectivos metadados, posteriormente disponibilizados pelo Professor Thiago Faleiros.

Após uma busca extensiva na literatura mais recente, detalhada na seção II, foi elaborado o método proposto, que consistirá na aplicação de um arquitetura *encoder-decoder* para definir um modelo *sequence-to-sequence*, também chamado de *Seq2seq* [4]. Na etapa de *encoder*, serão implementadas Redes Neurais Recorrentes (RNN, no inglês), mais especificamente uma LSTM bi-direcional que será treinada em cima da base de discursos e a matriz gerada por um modelo *Word2Vec* [5] também com o vocabulário completo de discursos e resumos. Na etapa do *decoder*, também será utilizada uma LSTM que será inicializada com os vetores de contexto do *encoder*, selecionados pelo mecanismo de *attention* e treinada com a base de resumos. A explicação dos métodos, bem como das decisões tomadas para a realização do projeto será detalhada na seção III.

Os resultados e performance do modelo serão expostos e comparados com os da literatura na seção IV e conclusões, limitações e trabalhos futuros serão comentados na seção V.

II. Revisão de Literatura

Nesta seção, será sintetizado o conteúdo de alguns artigos que foram utilizados como referência neste projeto, onde serão descritas as principais abordagens classificadas como relevantes para idealização do método proposto bem como uma análise de seus resultados.

A. Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks, 2018

Khatri et al. [6] utiliza técnicas extrativas e abstrativas para a realização de uma sumarização automática em descrições de produtos no *e-Bay* e compara seus resultados. Duas contribuições relevantes do trabalho são: a utilização de RNNs junto a uma abordagem *Seq2Seq* para enriquecer os modelos com o contexto extraído dos textos; a construção de vetores *document-context* para extrair os resumos aproximados que serão usados como rótulo em um modelo de aprendizagem semi-supervisionada.

Os resultados chegam a conclusão de que o uso da primeira técnica melhora os resultados do modelo e que o uso da segunda não mostra uma diferença significativa dos resumos gerados por humanos.

B. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond, 2016

Já neste artigo, Nallapati et al. [7] os autores utilizam um modelo já desenvolvido para tradução automática e o aplicam para sumarização automática abstrativa. O modelo é uma RNN *encoder-decoder* também com uma abordagem *Seq2Seq*, mais especificamente uma GRU (*Gated Recurrent Unit*) com o *attention mechanism* e bi-direcional, implementado para permitir a construção de um resumo baseado no contexto de palavras e de sentenças como um todo.

Os pesquisadores então propõem uma série de adaptações e adições a este modelo, com o propósito de resolver problemas específicos da sumarização, como um texto muito grande, a presença de palavras raras ou um vocabulário muito grande. Nas duas bases de dados testadas, o uso dos novos modelos tiveram resultados significativamente melhores que seus antecessores.

C. Abstract Text Summarization with a Convolutional Seq2seq Model, 2019

Aqui Zhang et al. [8], vindo de encontro as literaturas anteriores, é usada um modelo *Seq2Seq* baseado em modelos CNNs (*Convolutional Neural Networks*) hierárquicos. RNNs dependem de passos anteriores para se calcular o passo atual durante seu treino e isso dificulta sua paralelização, fato que é possibilitado pela CNN. Assim como no artigo [7], são implementadas técnicas para se resolver os desafios apresentados por textos muito grandes e com palavras raras.

Devido a arquitetura do método proposto permitir a paralelização de vários processos no treino do modelo, os pesquisadores verificaram uma melhora de performance significativa e consiste em relação às outras alternativas.

D. Attention Is All You Need, 2017

Neste artigo, apesar do objeto de estudo ser sobre a tarefa de tradução automática, Vaswani et al.[9] propõe uma abordagem muito inovadora e que poderia ser replicada de forma análoga à tarefa de sumarização automática. Ao também identificar o problema da dependência da RNN em camadas anteriores, ele desenvolve uma nova solução que exclui a necessidade de modelos RNN ou CNN, e se baseia completamente no *attention mechanism*, chamando-o de *Transformer*.

Comparando com as outras abordagens sequenciais, o *Transformer* se mostra com resultados levemente melhores mas com um custo de treino significativamente maior, criando um novo estado da arte para a tradução automática.

III. Método Proposto

Está seção explicitará o método proposto neste projeto. Nela será comentada como foi feito o pré-processamento do dados que posteriormente foram necessários para a construção do modelo. Por fim, serão fundamentadas as decisões acerca da escolha do modelo, bem como uma explicação sobre sua arquitetura e suas camadas.

A. O pré-processamento dos dados

A base foi construída a partir da extração de uma série de dados relacionados a discursos de senadores do Brasil. Estes dados estavam armazenados em um arquivo *.pickle* e divididos em: o texto do discurso em si, informações de identificação do pronunciamento e informações de identificação do senador que proferiu o discurso. A partir daí foram extraídas apenas as informações relevantes à tarefa de sumarização automática que consistiam no texto original identificado com o nome "Conteudo" e na sumarização deste conteúdo nomeada "TextoResumo" dentro do objeto "IdentificacaoPronunciamento". Não se sabe o contexto de como e nem por quem foi feito esse resumo mas, a partir de uma análise por inspeção de alguns documentos, ele foi identificado como válido para ser usado como rótulo do modelo. A base de dados possui 74.167 amostras válidas, ou seja, com textos únicos e seus respectivos resumos.

A partir daí então, o texto foi tratado de forma que os caracteres especiais, pontuações e números foram removidos bem como as abreviaturas comuns no linguajar político foram expandidas (*i.e* V. Exa. para Vossa Excelência, Sra. para Senhora, Exma. para Excelentíssima). Apenas no texto do discurso completo foram removidas as palavras que não trazem significado para o texto, chamadas de *stopwords* e as palavras restantes de uma ou duas letras. O motivo desta transformação não ser aplicada também a resumo é para tentar trazer uma maior fluidez ao texto predito pelo modelo. Outrossim, é preciso adicionar a estes textos de resumo marcadores especiais de começo e fim (*i.e.* "_START_" e "_END_") para auxiliar futuramente na condição de parada da sumarização.

A partir do corpo de texto uniformizado foi realizado processo de *tokenização* de suas palavras, ou seja, a transformação de cada sequência de palavras presentes nos discursos e seus respectivos resumos em uma sequência de números,

chamados de vetores, para que o modelo possa interpretá-los matematicamente. Este vetor é construído de forma que cada palavra tem um número que a identifica unicamente. No final desta conversão, os vetores tem tamanhos distintos, correspondendo ao número de palavras na sequência original. Para que tenham um tamanho fixo e igual entre todos, os menores são preenchidos com zeros e os maiores são truncados.

Como a técnica abstrativa da sumarização de texto está diretamente conectada com extração de sentido do texto, foi necessário também a utilização de uma técnica que não só fizesse a *tokenização* mas também extraísse um significado numérico das relações entre as palavras. Por isso, foi utilizado o modelo *Word2Vec* [5] que já é conhecido por resolver esse problema. Mais especificamente, foi usado o algoritmo CBOW (*Continuous Bag-of-Words*) por ser mais eficiente em corpos de texto grandes e ter uma abordagem onde a partir de um **contexto** quer se predizer a **palavra**. Este modelo é treinado com o vocabulário dos discursos e resumos para criar uma representação vetorial de cada palavra do vocabulário que capturam os significados entre as palavras e que futuramente serão utilizados como um peso no vetor de entrada do modelo.

B. O modelo

Modelos de Redes Neurais tradicionais são muito bons para tarefas de classificação pois conseguem generalizar regras que levam uma determinada entrada para uma saída especifica dentre as várias saídas possíveis e conhecidas. No caso de tarefas de tradução e sumarização automáticas, as entradas são uma sequência de palavras em que sua ordem tem bastante importância e, dependendo de sua posição e contexto no texto, podem ter significados diferentes e consequentemente classificações diferentes. Além disso o tamanho da saída deixa de ser conhecido, pois esta também precisa ser uma sequência de palavras.

Por isso, a abordagem escolhida para a resolução do problema foi a construção de um modelo *Seq2seq* similar ao proposto por Sutskever et al. [4], mas com algumas modificações. Este tipo de arquitetura é comumente formado por dois componentes principais para que, a partir de uma entrada de uma sequência de tamanho n possa se gerar uma saída de outra sequência de palavras de tamanho m, e no caso da tarefa de sumarização temos $m < n$.

Os componentes são chamados de *encoder* e *decoder* e podem ser facilmente implementados com a ajuda de RNNs, pois este tipo de rede neural permite entradas de tamanho variável e no processamento de uma palavra da sequência, consegue persistir a saída do processamento da palavra anterior, chamado de *hidden state*, e levar isto em consideração no cálculo corrente, simulando uma ”memória” no sistema.

Entretanto em RNNs comuns esta ”memória” não é suficiente para se extrair o significado e contexto de frases e textos longos ($> 15palavras$). Por isso no modelo desenvolvido é utilizada uma RNN especial, a LSTM [10], já que estas são especializadas em lidar melhor no ato de extrair e manter o significado geral da sequência de palavras independente de seu tamanho, além de solucionar problemas já conhecidos de

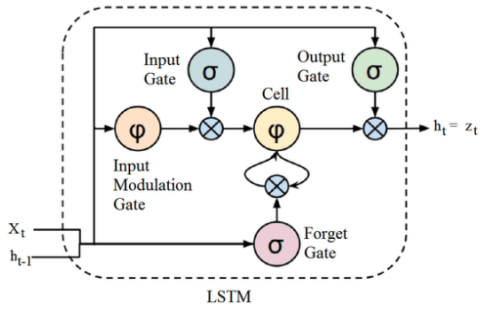

Fig. 1. Estrutura principal de uma LSTM, a célula. (Imagem acessada em Dezembro, 2020: http://deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/)

RNNs como o *exploding gradient* e *vanishing gradient*. Em sua estrutura, chamada de **célula** e demonstrada na figura 1, existem portas, ou *gates*, que controlam o fluxo de informações onde, além dos comuns *input* e *output*, temos também o *forget gate*, que decide quais informações continuam importantes e quais devem ser descartadas para as próximas células.

A LSTM por definição só propaga seus estados, os *hidden states*, agora selecionados pelo *forget gate*, para as células a sua direita na sequência de palavras. Isso significa que o contexto aprendido pelo modelo é sempre relativo às palavras passadas. Como será explicitado na seção IV, os discursos da base de dados têm a característica de serem muito prolixos e onde muitos protocolos iniciais são seguidos até que o assunto principal seja mencionado. Considerando esta situação, foi implementada um LSTM bi-direcional, inspirada na bi-direcionalidade do *Bidirectional Encoder Representations from Transformers* (BERT) [11]. Nesta implementação o modelo é configurado para guardar contexto tanto do passado, com uma LSTM normal, como o contexto do futuro, com uma LSTM configurada para executar a sequência na ordem inversa com a expectativa de que isto enriqueça a captura do contexto da sequência.

Na fase de treinamento da arquitetura *encoder-decoder*, desenhada na figura 2, o *encoder* processa, a cada intervalo de tempo, uma palavra $(x_1, x_2, ...x_n)$ de toda a sequência de entrada, no caso um discurso de um senador. O estado inicial (h_0) das LSTMs é zero. Em cada intervalo de tempo i, é calculado o valor atual da célula a partir de x_i, chamado de *cell state* (c_i). No *forget gate* (f), o resultado de c_i e o resultado do processamento imediatamente anterior ao seu, ou seja o *hidden state* h_{i-1} são multiplicados por matrizes de peso (W e U) e somados um *bias* (b), onde o valor é selecionado por uma função sigmóide (σ), passo explicitado na fórmula 1 e esboçado na figura 1.

$$f = \sigma(Wx_i + Uh_{i-1} + b) \tag{1}$$

No final da sequência, temos o último estado (c_n e h_n). chamado de vetor de contexto, pois nele é esperado que estejam guardadas as informações de contexto e sentido de um

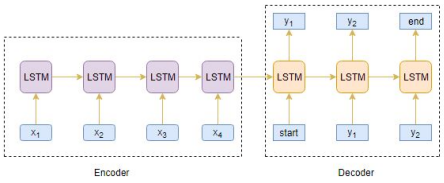

Fig. 2. Representação da fase de treinamento da arquitetura de modelo *encoder-decoder* (Imagem acessada em Dezembro, 2020: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/)

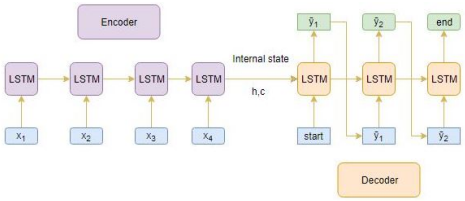

Fig. 3. Representação da fase de inferência da arquitetura de modelo *encoder-decoder* (Imagem acessada em Dezembro, 2020: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/)

determinado discurso. As duas LSTMs rodam independentes uma da outra e no final seus vetores de contexto são somados.

Este vetor de contexto é utilizado para inicializar o estado inicial (h_0) do *decoder*, que seguirá o mesmo processo do *encoder*, mas agora utilizando a sequência de palavras alvo como entrada, ou seja, a dos resumos dos discursos. Intuitivamente o *decoder* é treinado para predizer a próxima palavra com as informações de contexto do texto original. Aqui se fazem úteis os marcadores especiais de início e fim , pois para se predizer um resumo basta se inserir o marcador de início ao modelo e iterar seu processamento até que ele retorne o marcador de fim.

É justamente isto que acontece na fase de inferência da arquitetura, demonstrada na figura 3. O *encoder* processa toda a sequência com sua LSTM para se calcular o vetor de contexto, que é passado ao *decoder*, onde é inserido o marcador de início de frase no primeiro intervalo de tempo. A saída desta LSTM será a palavra com a maior probabilidade de ser a próxima palavra e está será utilizada como palavra de entrada da LSTM no próximo intervalo de tempo. Este processo é repetido, atualizando o estado da célula com a saída do *hidden state* anterior, até que a palavra de saída seja o marcador de fim.

Todavia, é importante notar que nesta arquitetura todo o contexto da sequência é resumida a um vetor tamanho fixo, o anteriormente citado vetor de contexto. Para sequências muito grandes, que é o caso dos discursos, este vetor não é suficiente para armazenar todas as informações importantes e repassá-las ao *decoder*. Para se resolver este desafio, foi implementado no modelo o *attention mechanism* proposto por Bahdanau et al. [12]. A intuição deste mecanismo é inserir no *decoder* um modo de decidir quais partes da sequência de entrada importam mais para definir a sequência de saída e assim focar nelas, ao invés de ter que processar a sequência inteira. Para isto, essa camada adicional faz uma soma ponderada de todos os *hidden states* do *encoder*, onde os pesos são definidos pela probabilidade condicional de uma palavra do entrada predizer corretamente uma da saída do *decoder*. Esse vetor resultado é calculado em uma função *softmax* e adicionado ao vetor de contexto que vai inicializar *decoder*.

IV. Resultados Experimentais

A. Análise da base

O maior desafio apresentado pela base de 74.167 discursos válidos foi o tamanho médio dos discursos e o extenso vocabulário. Como a base é formada pela transcrição de discursos no Senado Federal, muitas vezes os pronunciamentos são extensos e fogem de seu assunto principal, além do uso palavras rebuscadas que não seguem o léxico coloquial.

Ao todo são 151.430 palavras no vocabulário, sendo as mais frequentes 'não', 'senhor', 'senador', 'presidente', excelência', 'ser', 'governar', 'brasil', 'ano', 'brasileiro', o que é o esperado e está dentro do jargão político. As que aparecem menos de 15 vezes foram desconsiderados para a construção do *Word2Vec*, que tem um vocabulário final de 43.731 palavras com dimensão do vetor igual a 100.

Para se testar o quão bem o *Word2Vec* está funcionando, foram testadas alguma palavras que são específicas do jargão político. O vetor da palavra 'PT' que significa Partido dos Trabalhadores, sigla do partido que alguns senadores, tem como vetores mais similares 'partido', 'petistas', 'democratas', 'disputa', 'candidata', 'desfiliar', que fazem total sentido dentro do contexto do objeto de estudo e asseguram que nosso *word embedding* conseguiu capturá-lo.

O número máximo de palavras em um discurso é de 18.114 e o mínimo é 2 palavras. O tamanho médio é de 762 com desvio padrão de 687, ou seja, um conjunto muito esparso. Já para o resumo do discurso temos 225 palavras como máximo, 2 como mínimo, com média de 17 palavras e desvio padrão de 9. Os *outliers* foram removidos e base final é composta por todas as amostras entre o 1-percentil e 99-percentil da variável que correspondia ao número de palavras total do discurso, finalizando com um total de amostras de 72.675 amostras. Como explicado anteriormente na seção III, todos estes textos são convertidos em vetores e devidamente preenchidos com zeros, caso necessário.

Antes de enviar as entradas para o modelo, 80% da base é separada para treino do modelo e 20% é separada para o futuro teste de sua eficiência.

B. Configurações do modelo

Como o modelo precisa de um tamanho fixo de entrada no *encoder* foi inicialmente escolhido como tamanho de corte o valor do 90-percentil, 1524 palavras. Para o resumo o valor do 90-percentil é de 29 palavras, que será o tamanho dos vetores de saída. Entretanto, esta configuração, apesar de ser ideal, não foi a escolhida pois gerava na fase de treinamento muitos parâmetros para serem calculados e devido

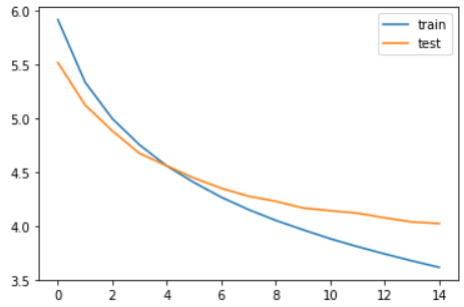

Fig. 4. Gráfico representando a evolução da variável *loss* (eixo y) por época (eixo x), para as massas de treino (em azul) e teste (em laranja)

a limitações de *hardware* foi escolhido o valor de 50-percentil para o discurso, sendo 610 e 15 palavras respectivamente. Essa decisão reduz o número de parâmetros de mais de 50 milhões para aproximadamente 20 milhões.

A LSTM foi inicialmente configurada para entregar em sua saída um vetor de tamanho 1524, o mesmo tamanho do vetor de entrada, mas teve que ser redimensionada com tamanho 200 pelo mesmo motivo, as limitações de *hardware*. Esta redução de dimensionalidade diminuiu consideravelmente a capacidade do modelo de extrair melhor o contexto dos textos originais.

O modelo foi desenvolvido utilizando a API *Keras* do *Tensorflow* [13] e treinado na plataforma *Google Colabs* por ela fornecer de forma gratuita uma GPU, um tipo especializado de processador especializado em processamento paralelo. Apesar do processamento de modelos RNN ser sequencial, podemos utilizar uma biblioteca chamada *cuDNN* para GPUs que a partir de uma série de técnicas pode acelerar em até uma ordem de grandeza a etapa de treinamento e inferência do modelo [14]. Após as configurações de parâmetros necessárias essa biblioteca foi implementada no modelo.

Na fase de treinamento, o modelo foi configurado para treinar 15 épocas, levando em torno de 7 minutos de tempo de processamento por época. Sem as reduções de dimensionalidade e o uso da GPU, seria virtualmente impossível o treino do modelo, já que anteriormente a essas medidas, o modelo levava até 12 horas por época e consumia toda a memória da máquina utilizada ($>$ 16 gigabytes). A figura 4 demonstra o resultado do modelo pela evolução da *loss*, parâmetro calculado utilizando o número de exemplos de treino e teste que o modelo acerta comparados ao que ele erra, que procuramos minimizar a cada época. A curva do teste, a partir da oitava época está levemente acima da curva de treino, o que indica uma tendência de sobreajuste do modelo para as próximas épocas, mas que devido sua baixa intensidade, não se mostrou problemática.

Os resumos são então preditos para toda a base de teste, na etapa de inferência e as métricas são calculadas para cada um deles. A inferência é a etapa que teve o maior tempo de processamento, aproximadamente 2 horas e 30 minutos.

TABLE I
EXEMPLOS DE RESUMOS GERADOS

Textos dos Resumos (**O**: resumo original, **P**: resumo predito)
Exemplo 1: **O**: consideracoes importancia avanco biotecnologia **P**: consideracoes a importancia politica agricola e o pais
Exemplo 2: **O**: interpelacao ministro desenvolvimento industria e comercio senhor sergio silva amaral **P**: interpelacao senhor ministro relacoes exteriores senhor celso lafer
Exemplo 3: **O**: crise estrutural brasileira **P**: consideracoes a crise economica pais
Exemplo 4: **O**: limite participacao investimento estrangeiro aviacao civil conforme estudo contratado dem e criticas modelo atual **P**: consideracoes a crise economica pais

C. Métricas

Para se avaliar a performance do modelo foram utilizadas duas técnicas. A primeira foi uma análise qualitativa, a partir de uma inspeção visual em aproximadamente 30 amostras. Nas amostras analisadas o resumos gerados conseguem capturar bem a essência do texto original e alguns padrões de comportamento estão demonstrados, ainda tokenizados, na tabela I. O Exemplo 1, por exemplo, sugere que o modelo conseguiu manter o sentido de ”avanco biotecnologia” no original para ”politica agricola e o pais”.

O segundo padrão, indica o caso de nomes próprios e instituições. Normalmente o modelo prediz corretamente sua existência e posição na frase mas erra o nome ou instituição. No Exemplo 2 podemos ver que ele confundiu a instituição ”[ministerio] desenvolvimento industria e comercio” com ”[ministerio] relacoes exteriores” e a pessoa ”sergio silva amaral” com ”celso lafer”.

Outro padrão observado é que a maioria dos resumos preditos começam com frases como ”considerações a”, ”criticas a”, ”defesa a”, que é um início de sentença comum visto resumos gerados por humanos, e podemos inferir que esta característica que foi herdada e replicada pelo modelo. O resumo do Exemplo 3 é um exemplo disto.

A maioria de casos de resumos de má qualidade são as vezes que o modelo constrói um resumo generalista demais que acaba informando ao leitor pouco ou nenhum contexto do texto original. Este caso esta demonstrado no Exemplo 4, onde apesar de correto o resumo não traz a informação necessária que esta presente no resumo feito por humanos.

A segunda análise foi quantitativa, calculando-se duas métricas para sumarização de textos observadas em outros estudos. A primeira é a métrica mais clássica da literatura, a ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) [15], que mede quantos *n-gramas*, ou sequência de n palavras, do resumo escrito pelo modelo correspondem no resumo de referência, escrito por humanos. Desta correspondência

TABLE II
RESULTADOS PARA O MODELO PROPOSTO

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
Projeto	0.18	0.06	0.17	0.68
words-lvt2k-1sent [7]	0.36	0.17	0.32	-
AC-RNN [6]	0.33	0.31	0.27	-
Paraschiv et al. [17]	0.42	0.55	0.43	0.3294

podem-se calcular a precisão e revocação, e como forma de computar os dois em uma única métrica, aqui é usado o *F1-score*, indicado na fórmula 2. Para cada amostra foi calculado o valor de três variações desta métrica, o ROUGE-1 (para correspondência de unigramas), ROUGE-2 (para correspondência de bigramas) e ROUGE-L (para a maior sequência correspondente).

$$F_1 = 2 \cdot \frac{\text{precisao} \cdot \text{revocacao}}{\text{precisao} + \text{revocacao}} \tag{2}$$

Um problema evidente da ROUGE é que nem humanos diferentes fazem resumos com as mesmas palavras, ou seja, seu valor máximo é impossível de ser alcançado na realidade e pode acabar levando a interpretações erradas quando comparamos a mesma métrica em metodologias de treinamento e bases de dados diferentes. Por isso, também é utilizada uma outra métrica, mais recente, e que lida melhor com esse desafio, o BERTScore [16]. Ele aplica uma comparação de cossenos nos *word embeddings* pré-treinados do BERT para português, o que significa que ele compara os **significados** entre o resumo predito e o de referência e não só a correspondência de palavras. Assim como a ROUGE, conseguimos calcular o *F1-score* a partir da precisão e e revocação gerada pelo modelo, indicada na fórmula 2.

Os *F1-scores* das métricas foram calculadas para a base de teste e se encontram na tabela II, bem como uma comparação em relação aos resultados encontrados em alguns modelos similares da literatura relacionada. Como o BERTScore foi publicado ainda este ano, foi encontrada apenas uma publicação que utiliza um modelo similar ao deste projeto e o utiliza como métrica.

Pode se notar que as métricas ROUGE estão com valores bem abaixo do que os outros modelos. Isso pode ser explicado pelo fato própria característica da construção de resumos de forma abstrativa onde o objetivo não é necessariamente a correspondência de palavras e sim a do sentido. Quando o BERTScore é comparada com as demais nota-se que ela demonstra melhores resultado, indicando que o texto gerado, apesar de não possuir as mesmas palavras que o resumo original conseguiu capturar bem seu sentido.

V. CONCLUSÃO

Neste projeto foi proposta uma aplicação da arquitetura *encoder-decoder* para modelos *sequence-to-sequence* para realização da geração automática de resumos de maneira abstrativa, ou seja, contextual. Essa abordagem foi implementada com o uso de RNNs, a fim de se aprender o contexto das longas sentenças presentes nos textos completos e utilizar este conhecimento para a construção do texto resumido. LSTMs bi-direcionais foram utilizadas para se armazenar esse contexto do passado e futuro de cada texto a longo prazo, que foram em seguida selecionadas, por um *attention mechanism*, considerando sua relevância para a construção de um resumo objetivo.

Os resumos preditos pelo modelo foram construídos em uma etapa de inferência e avaliados por inspeção visual e por duas métricas, a ROUGE para correspondência de palavras e o BERTScore para a correspondência de contexto entre o resumo gerado por humanos e o gerado pelo modelo. A análise das métricas sugere que apesar do modelo não conseguir utilizar as mesmas palavras que o resumo original, ele consegue manter o sentido presente neste.

Entretanto, algumas falhas foram observadas e pretende-se adicionar em trabalhos futuros novas abordagens, na tentativa de consertá-las. Para os resumos muito generalistas, pode-se tentar a aplicação do *attention mechanism* local proposto por Luong et al. [18] que seleciona melhor a relevância do contexto em textos maiores. Esse contexto também pode ser enriquecido na fase de treinamento do *encoder* com a implementação de uma nova direcionalidade na LSTM, um processamento a nível de frases ou parágrafos inteiros dentro da sequência original, como sugerido por Nallapati et al. [7].

Outro ponto que pode agregar ao modelo com uma melhor extração da relação entre as palavras e a diminuição vieses de estereótipos e preconceitos é a utilização do *word embedding* pré-treinado *ConceptNet Numberbatch* [19], que foi construído baseado em nos famosos *Word2Vec*, utilizado no projeto, e *GloVe*.

A adição de uma camada de NER (*Named-Entity Recognition*) no *decoder* poderá melhorar a identificação das entidades corretas e, junto a uma camada de PoST (*Part-of-Speech Tagging*), pode trazer uma melhora na fluidez dos resumos gerados.

Por último, houve a redução de dimensionalidade do vetor de contexto entre o *encoder* e *decoder* que gerou uma perda significativa de informação relevante. Foi observado na literatura que os pesquisadores chegam a treinar seus modelos em *clusters* de até 8 GPUs em quase 4 dias de tempo de processamento [9][7]. Por isso para lidar com esta limitação de hardware é possível tanto adquirir uma melhoria no hardware disponível para este projeto como procurar implementar técnicas com menores custos computacionais, como o uso de *Transformers* [9] ou da substituição das RNNs por CNNs [8].

REFERENCES

[1] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[2] Paula Christina Figueira Cardoso. *Exploração de métodos de sumarização automática multidocumento com base em conhecimento semântico-discursivo*. PhD thesis, Universidade de São Paulo, 2014.

[3] Senado Federal dados abertos. https://www12.senado.leg.br/dados-abertos. Accessed: 2020-11-16.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Chandra Khatri, Gyanit Singh, and Nish Parikh. Abstractive and extractive text summarization using document context vector and recurrent neural networks. *arXiv preprint arXiv:1807.08000*, 2018.

[7] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[8] Yong Zhang, Dan Li, Yuheng Wang, Yang Fang, and Weidong Xiao. Abstract text summarization with a convolutional seq2seq model. *Applied Sciences*, 9(8):1665, 2019.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[14] Jeremy Appleyard, Tomas Kocisky, and Phil Blunsom. Optimizing performance of recurrent neural networks on gpus. *arXiv preprint arXiv:1604.01946*, 2016.

[15] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[16] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.

[17] Andrei Paraschiv. Upb at germeval-2020 task 3: Assessing summaries for german texts using bertscore and sentence-bert.

[18] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[19] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. pages 4444–4451, 2017.