Exercício Programa 3

Felipe de Lima Peressim[1]

MAC5725 - Linguística Computacional

Professor: Dr. Marcelo Finger

São Paulo

2020

[1] NUSP: 11823558. E-mail: felipe.peressim@ime.usp.br

Sumário

1 Introdução

O presente relatório se constitui de quatro seções principais: a segunda seção descreve o preprocessamento e a preparação dos dados comum as duas arquiteturas de redes neurais propostas no enunciado. Na terceira seção, são descortinados os pormenores concernentes a arquitetura *Encoder-Decoder BiLSTM*. Inicialmente são descritas as etapas de preparação e preprocessamento dos dados específicas para a arquitetura em questão; em seguida são apresentados as configurações e os hiperparâmetros utilizados; por conseguinte, a etapa de treinamento e avaliação do modelo são detalhados e os resultados obtidos durante tais fases são discutidos.

Na quarta seção, de maneira semelhante a terceira seção, a mesma abordagem é utilizada para discutir as particularidades e os resultados concernentes à arquitetura *BERT*. Por último, na quinta seção, os resultados obtidos por ambos os modelos são sumarizados para se fazer uma discussão e uma reflexão de maneira geral acerca dos objetivos atingidos através dos experimentos que foram realizados.

Todos os experimentos foram realizados no ambiente de desenvolvimento *Google Colaboratory* e os modelos foram treinados com o auxilio de uma *GPU NVIDIA Telsa T4* com *16GB* de *ram* disponibilizada pelo Google.

2 Preprocessamento

A etapa de preprocessamento comum as duas arquiteturas se deu da seguinte forma: Na primeira etapa, após carregar o conjunto de dados com auxílio da biblioteca *Pandas*, todas as amostras de dados duplicados com respeito à variável *text review* foram removidas. Na sequência todas as amostras com sentenças das variáveis *text review* e *title review*, nas quais o comprimento é inferior a três caracteres, também foram removidas.

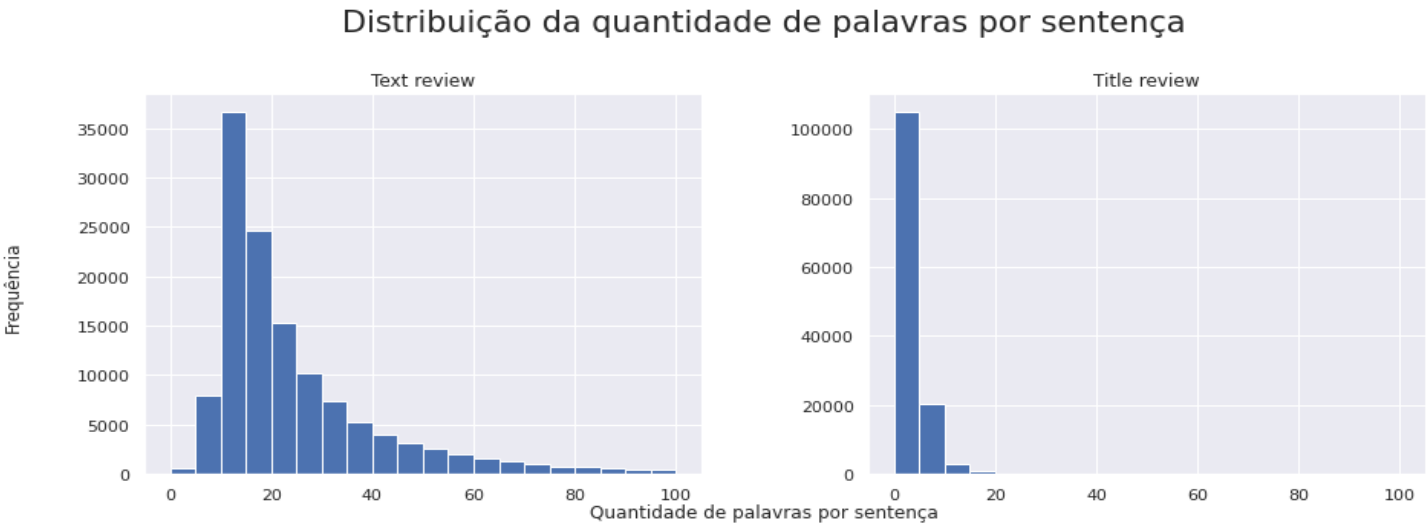
Em seguida, as amostras passaram pelas seguintes transformações:

1. Todos os caracteres foram convertidos para minúsculas e os espaços desnecessários à direita e à esquerda foram removidos;
2. a ocorrência de dígitos nas sentenças foram removidas;
3. a repetição de pontuações foi substituída por uma única instância;
4. um espaço foi criado entre uma palavra e a pontuação que a segue;
5. contrações informais foram substituídas por sua palavra formal equivalente; *e.g* vc : você; tbm : também; etc;

6. por último, símbolos especiais foram removidos.

Depois de se realizar a etapa de limpeza descrita, para se determinar a quantidade máxima de palavras em cada cada uma das váriaveis *text review* e *title review* respectivamente, realizou-se uma análise da distribuição da quantidade de palavras por sentença nessas variáveis, obtendo-se assim o histograma apresentado na Figura 1 a seguir:

Figura 1 – Distribuição da quantidade de palavras nas sentenças das variáveis *text review* e *title review*.



A etapa de preprocessamento para alimentar o modelo *Encoder-Decoder BiLSTM*, foi realizada sobre o conjunto de dados completo da *B2W Reviews*, ou seja, o conjunto antes da etapa de preprocessamento contava com 132374 amostras de dados, e, após tal etapa ficou com 122685 amostras de dados, tais quais o histograma das distribuições apresentado na Figura 1 foi construído. Por outro lado, para alimentar o modelo *BERT*, apenas 25% dos dados do conjunto total, foram selecionados de maneira aleatória e carregados, ocasionando em um total de 33093 amostras de dados. Assim, visto que esses 25% vem da mesma distribuição de dados, e os histogramas da distribuição de sentenças se mostraram semelhantes, por estes motivos apenas um deles é apresentado aqui.

Desta forma, após analisar as distribuições da quantidade de palavras por sentença, definiu-se que as sentenças da variável *text review* não teriam mais do que 64 palavras, e que as sentenças da variável *title review*, não teriam mais do que 15 palavras, pois em ambos os casos, a distribuição das palavras com no máximo as quantidades mencionadas, constituem mais de 80% da distribuição total.

Após realizar as etapas supramencionadas, ambos os modelos *Encoder-Decoder BiLSTM* e *BERT* passaram por uma etapa adicional de preparação dos dados antes de se realizar a etapa

de tokenização, no entanto, essas etapas serão descritas e justificadas nas seções seguintes visto que os procedimentos adotados nessa etapa adicional utilizam diferentes estratégias em cada um dos modelos.

3 Encoder-Decoder BiLSTM

3.1 Estratégia de implementação e etapa adicional de preparação e preprocessamento dos dados

A etapa de preparação e preprocessamento adicional concernente ao modelo desta seção, consistiu inicialmente em adicionar tokens especiais de inicio e fim de sentença, $<start>$ e $<end>$ respectivamente. Tais tokens especiais são adicionados à sequência de saída do modelo, ou seja, na variável *title review*. A sequência alvo é desconhecida durante a decodificação da sequência de teste. Então, começamos a prever a sequência alvo passando a primeira palavra para o decodificador, que seria sempre o token $<start>$. E o token $<end>$ sinaliza o fim da frase. Dessa forma, o modelo aprende a decodificar onde a sentença começa e onde termina.

Posteriormente a esta etapa, as amostras de dados são tokenizadas com auxilio da classe *Tokenizer* do *Keras*. Nesta fase, o tokenizador é configurado apenas para transformar as sentenças em sequências numéricas com índices das palavras, ou seja, nenhuma transformação de de preprocessamento adicional é realizada. Para tanto, o vocabulário é construído pelo tokenizer com base nas amostras de dados que o alimentam. Para o experimento em questão dois tokenizadores foram adaptados aos dados das váriaveis *text review* e *title review*, resultando em vocabulários com 49145 palavras e 13622 palavras respectivamente.

Por último, as sequências resultantes foram preenchidas com o token especial *PAD*, com o método *post* e truncadas com os limites de quantidade de palavras configurados durante a etapa de preprocessamento anterior, citada na seção de preprocessamento. Após tais etapas o modelo *Encoder-Decoder BiLSTM* com atenção foi configurado e preparado para a etapa dos experimentos. A seguir tais configurações e os híper-parâmetros utilizados são descritos.

3.2 Descrição das configurações e dos híper-parâmetros utilizados

3.2.1 Descrição das configurações e dos dos híper-parâmetros do Encoder

As Tabelas 1 e 2 descrevem as configurações e hiper-parâmetros que foram configurados na camada de *Input*, *Embedding* e nas camadas de *foward* e *backward* da camada bidirecional, que constitui o encoder do modelo proposto.

O tamanho da entrada para a camada de *Input* exibido na Tabela 1, foi definido como 64, ou seja, a quantidade máxima de palavras por sentença, que foi definida para a variável de entrada do modelo - *text review*. A camada de embedding do Keras foi utilizada com o parametro *trainable* ativado, ou seja, os pesos da matriz vão sendo ajustados durante o treinamento do modelo. A dimensão da matriz foi configurada com a dimensão 49145 x 128, em que a primeira dimensão é o quantidade de palavras do vocabulário com base na variável *text review* e a segunda dimensão diz respeito a quantidade de pesos por palavra em que se estipulou para tal camada.

Input e Embedding - Encoder	
Input - Tamanho da entrada	64
Embedding Keras - Dimensão	49145 x 128
Embedding Treinable	True

Tabela 1 – Configuração da camada de Input e da camada de Embedding do Keras.

LSTM Bidirecional - Encoder	
Qtd células	64
Função de ativação camada oculta	Tanh
Função de ativação Recorrente	Sigmoid
Return State	True
Return Sequences	True

Tabela 2 – Configuração das camadas Foward e Backward da camada Bidirecional do Encoder.

O *encoder* do modelo foi construido com uma camada bidirecional LSTM, e, para ambas as camadas, *forward* e *backward* que a constituí, as seguintes configurações apresentadas na Tabela 2, foram as mesmas. A quantidade de células - 64 - constitui 50% do tamanho da segunda dimensão do *embedding*, pois, como o *hidden state* e o *cell state* são concatenados antes de serem usados pelo *decoder*, passam a ter a mesma dimensão de 128, que foi definida para os embeddings do *encoder* e do *decoder*.

As funções de ativação das camadas *forward* e *backward* não foram alteradas, ou seja, foram mantidas no padrão do artigo original *Tanh* e *Sigmoid*. Os parâmetros *return state* e *return sequences* foram marcados como *true*, assim, os resultados produzidos por tais camadas são concatenados e utilizados pelo *decoder* e pela camada de atenção.

3.2.2 Descrição das configurações e dos dos híper-parâmetros do Decoder

As Tabelas 3 e 4 descrevem as configurações e hiper-parâmetros que foram configurados na camada de *Input*, *Embedding* e na camada LSTM que constitui o *decoder* do modelo

proposto.

Input e Embedding - Decoder	
Input - Tamanho da entrada	None
Embedding Keras - Dimensão	13622 x 128
Embedding Treinable	True

Tabela 3 – Configuração da camada de Input e da camada de Embedding do Keras.

LSTM - Decoder	
Qtd células	128
Função de ativação camada oculta	Tanh
Função de ativação Recorrente	Sigmoid
Return State	True
Return Sequences	True

Tabela 4 – Configuração das camadas Foward e Backward da camada Bidirecional do Encoder.

O tamanho da entrada para a camada de *Input* exibido na Tabela 3, foi definido como *None*, ou seja, o Keras fica responsável por delinear o tamanho da entrada, baseado nas dimensões da matriz de saída do *encoder*. A camada de embedding do Keras foi utilizada com o parametro *trainable* ativado, ou seja, os pesos da matriz vão sendo ajustados durante o treinamento do modelo. A dimensão da matriz foi configurada com a dimensão 13622 x 128, em que a primeira dimensão é o quantidade de palavras do vocabulário com base na variável *title review* e a segunda dimensão diz respeito a quantidade de pesos por palavra em que se estipulou para tal camada.

O *decoder* do modelo foi construido com uma camada LSTM, que é constituída pelas configurações apresentadas na Tabela 4. A quantidade de células - 128 - compatível com a dimensão do embedding desta, que é compatível com a dimensão da saída do *encoder*, que é inferida pelo Keras.

As funções de ativação das camadas não foram alteradas, ou seja, foram mantidas no padrão do artigo original *Tanh* e *Sigmoid*. Os parâmetros *return state* e *return sequences* foram marcados como *true*, assim, os resultados produzidos por tais camadas são concatenados e utilizados na camada de atenção juntamente com a saída do *encoder*.

3.2.3 Camada de atenção

O mecanismo de atenção foi implementado como uma classe compatível com o Keras, ou seja, herdando atributos da classe *Layer* e implementando alguns mecanismos básicos necessários. A classe em questão implementa a atenção de Bahdanau, conforme descrita

no artigo original de [Bahdanau et al., 2014]. E o código foi baseado na implementação de [Ganegedara, 2020].

3.2.4 Camada densa

A saída do modelo foi configurada uma camada *Dense* envolvida por uma camada *Time Distributed*, com 13622 neurônios - tamanho do vocabulário do *title review* - e função de ativação *Softmax*.

3.3 Processo de inferência do modelo

O processo de inferência foi implementado de modo a se receber uma sentença e gerar um título para ela. Para isso, o texto em questão passa pelos processos de preprocessamento e tokenização descritos anteriormente, e então, os seguintes passos são realizados:

1. A sequência é codificada pelo *encoder* para se obter os estados internos para alimentar o *decoder*;
2. O token especial de inicio de sentença - $<start>$ - é passado como entrada para o *decoder*;
3. O *decoder* é utilizado com os estados internos para um único *timestep*;
4. A saída será o vetor de probabilidades, e a maior probabilidade que representa a próxima palavra é selecionada com *argmax*;
5. A palavra prevista pelo modelo é usada como entrada no *decoder* para o próximo *timestep*, de modo que a atualizar os estados internos com o *timestep* atual;
6. Os passos de 3 à 5 são repetidos até que o modelo gere o token especial de fim de sentença - $<end>$, ou, a sentença gerada atinja um máximo de 15 palavras - tamanho máximo estipulado durante a etapa de preprocessamento;
7. Por último a sequência é transformada em texto e retornada para o mensurar as pontuações com as métricas propostas no enunciado.

3.4 Treinamento e Validação

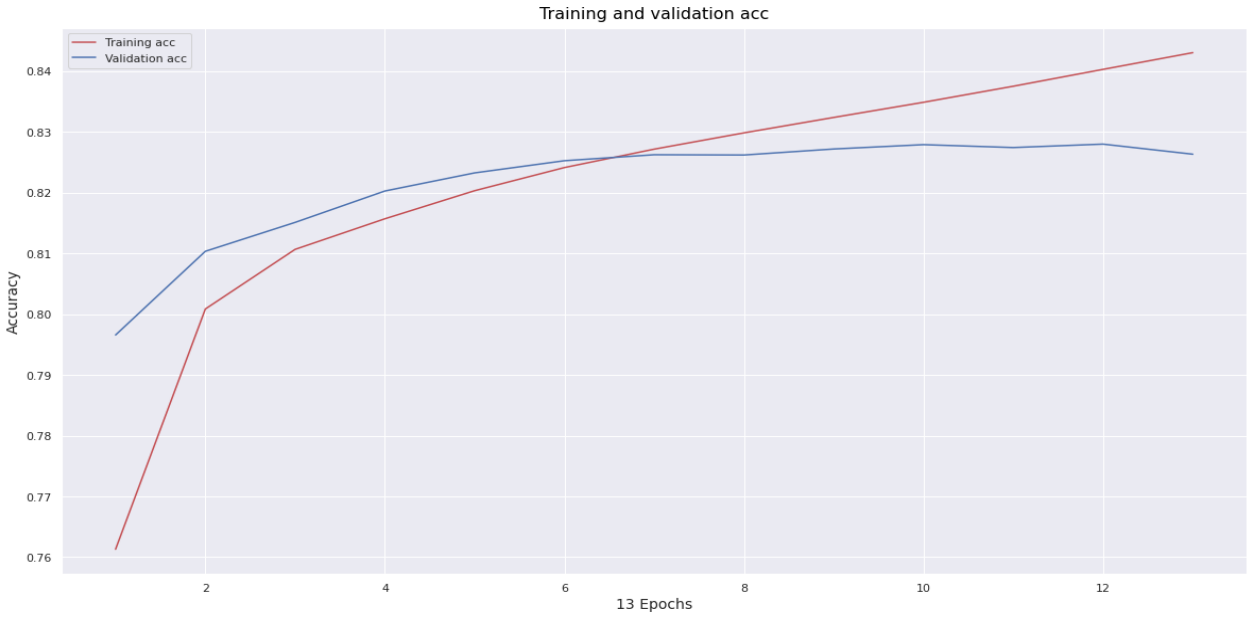
A função *loss* utilizada foi a *Sparse Categorical Crossentropy*, e o algoritmo Adam foi utilizado como função de otimização. Os parâmetros do Adam não foram alterados, manteu-se os padrões do Keras que segue o artigo original. O tamanho de cada lote (*batch size*)

utilizado durante a etapa de treinamento foi definido como 256 amostras por lote. Para o treinamento um total de 25 épocas foram estipuladas.

Para monitorar o desempenho do modelo, o *callback Early Stopping* foi configurado para monitorar o desempenho da *loss* com base no conjunto de validação. Especificamente, o *Early Stopping* foi configurado para interromper o treinamento do modelo quando não houver melhora de desempenho dentro de 2 épocas de treinamento. Por último, os dados utilizados para treinar, validar e testar o modelo, foram particionados na proporção 80% para o conjunto de treinamento; 10% para o conjunto de validação e 10% para o conjunto de testes. Assim, o modelo contou com um total de 103340 amostras no conjunto de treino; 12917 amostras no conjunto de validação e 12918 amostras no conjunto de teses.

Durante o treinamento do modelo a estratégia *teaching forcing* foi utilizada - os dados de saída são acoplados aos dados de entrada para retroalimentar o modelo. Tal estratégia refletiu no a acurácia exibida no desempenho do modelo, que é razoavelmente alta considerando a tarefa de geração de textos, que aqui estamos realizando, no entanto, conforme mencionado no enunciado, tal métrica não é confiável para esse tipo de atividade.

Figura 2 – Curvas de acurácia do modelo Encoder Decoder BiLSTM nos conjuntos de treino e validação.



A Figura 2 exibe o desempenho do modelo durante o processo de treinamento e validação do modelo. Durante tal etapa, a melhor acurácia média obtida no conjunto de treinamento foi 84,30% durante a época de número de 13. A melhor acurácia média no conjunto de validação foi de 82,80% durante a época de número 12. O treinamento começou a sofrer *overfitting* durante a época 7 e o treinamento foi interrompido pelo *callback* durante a época 13. Após está etapa o modelo foi testado no conjunto de testes e assim obteve uma acurácia

média de 82,50%. Mas, como essa medida não é confiável para a tarefa de geração de textos, outras métricas e metodologias foram utilizadas para avaliar o modelo no conjunto de testes, que são descritos na seção a seguir.

3.5 Testes e Resultados

Para se realizar os testes, além da acurácia média, duas métricas adicionais foram consideradas. As métricas escolhidas foram o BLEU (do inglês: *Bilingual Evaluation Understudy*) de [Callison-Burch et al., 2006], e o METEOR (do inglês: *Metric for Evaluation of Translation with Explicit ORdering*) de [Banerjee and Lavie, 2005], que foram utilizadas com auxílido da biblioteca de processamento de linguagem natural NLTK.

Também se realizou uma avaliação do texto gerado com auxilio de um ser humano. Tal avaliação foi realizada considerando-se 200 amostras de texto geradas pelo modelo e selecionadas ao acaso. O aluno e colega de pós-graduação Alexandre Locci foi quem realizou tal avaliação, atribuindo as pontuações 0, 0,5 e 1,0 conforme solicitado para este trabalho, assim doravante denominaremos a avaliação humana de medida-Locci.

3.5.1 BLEU

O BLEU implementado no NLTK testa de 1 até 4-grams as sobreposições do texto gerado em comparação com o original. Além disso, para se computar as pontuações foi necessário converter as 12918 sequencias do conjunto de texto para o formato de texto (*string*). Isso ocasionou em um tempo demasiado de 36 minutos para se calcular as pontuações. A tabela 5 exibe as estatísticas calculadas com as pontuações do BLEU nas amostras testadas.

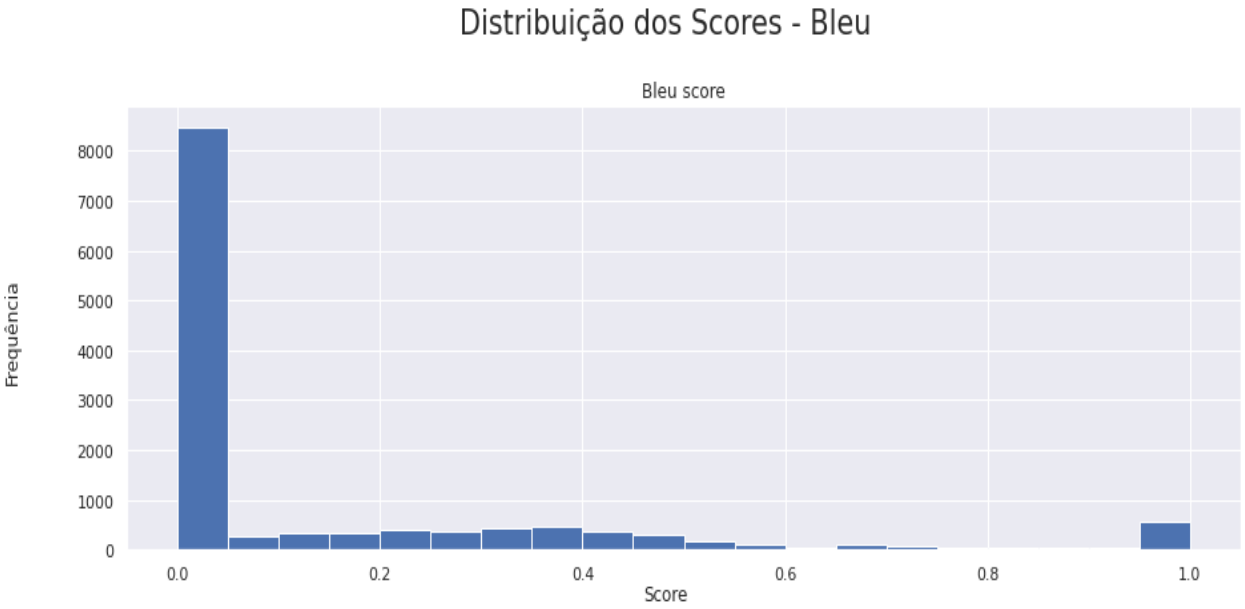
Estatísticas - BLEU - 12918 sentenças	
Minímo	0,0
Média	0,1459
Máximo	1,0
Desvio Padrão	0,2578

Tabela 5 – Estatísticas calculadas com as pontuações do BLEU para 12918 sentenças do conjunto de testes.

A melhor pontuação foi o valor máximo de 1, conforme exibido na tabela 5, o que de acordo com a Figura 3 significa que o modelo previu cerca de 500 títulos idênticos aos originais - cerca de 3,87% da distribuição. Por outro lado, a menor pontuação prevista - 0 (zero) - está dentro do grupo com maior frequência de pontuações, aproximadamente 8500 amostras dentro de tal faixa, conforme se pode observar na Figura 3, que se encontra dentro de um intervalo de pontuações entre 0 e 0,1, aproximadamente, o que representa 66% das

pontuações calculadas. Essa observação fica um pouco mais evidente ao se observar que a média das pontuações foi de 0,1459 com desvio padrão de +/- 0,2578.

Figura 3 – Distribuição das pontuações do BLEU nas 12918 sentenças.



Os resultados obtidos foram muito baixos, no entanto, não refletem a capacidade real de geração de títulos do modelo, que no geral foi muito boa e se mostrou cumprir com aquilo que foi postulado, mais adiante com a avaliação humana e com alguns exemplos de títulos gerados, ter-se-á uma ideia mais fiel da capacidade real do modelo.

3.5.2 METEOR

O METEOR também levou um tempo similar ao BLEU de 36 minutos para calcular as pontuações para todas as sentenças do conjunto de testes. Cabe ainda salientar, que parte da demasia no tempo também é devida ao tempo gasto em que o processo de inferência leva para prever e decodificar os títulos. A Tabela 6 a seguir apresenta os resultados obtidos calculando-se as pontuações do METEOR com auxílio da biblioteca NLTK no conjunto de testes.

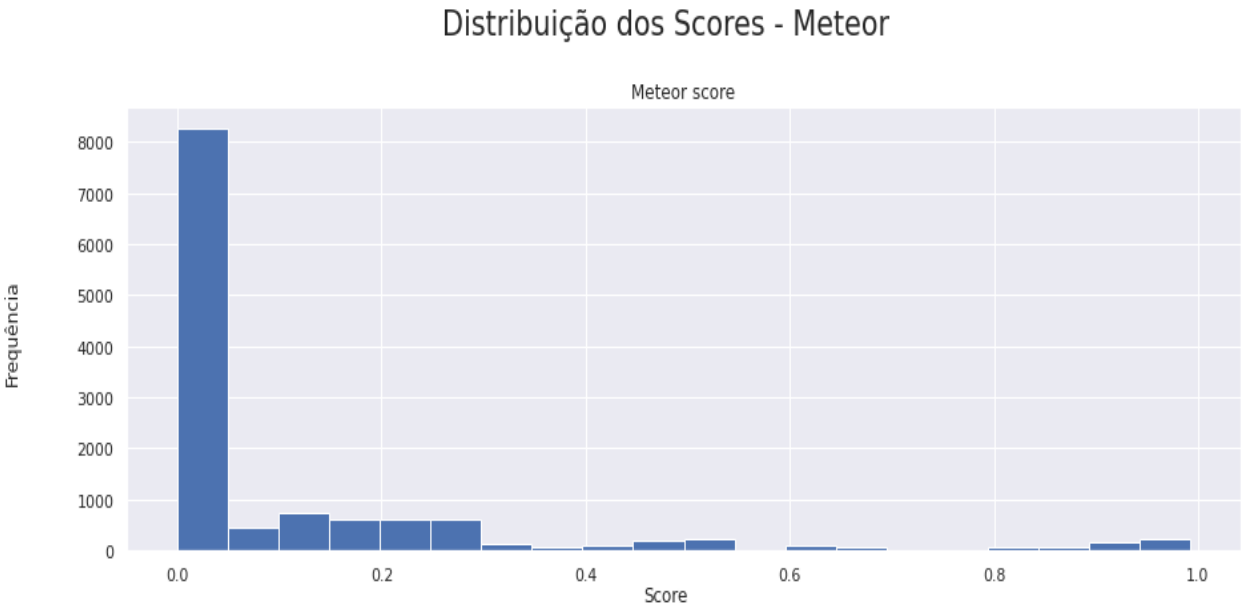
A maior pontuação foi de 0.9922, cerca de 2,3%, o que representa aproximadamente 300 amostras da distribuição, conforme se pode observar na Figura 4. Em contrapartida, a menor pontuação prevista foi 0 (zero), ou seja, de maneira semelhante ao BLEU, tem se a ocorrência títulos previstos em que as palavras formadoras da sentença diferem totalmente daquelas que formam o título original. Tal pontuação mínima se encontra dentro de uma faixa de valores

Estatísticas - METEOR - 12918 sentenças	
Minímo	0,0
Média	0,1182
Máximo	0,9922
Desvio Padrão	0,2249

Tabela 6 – Estatísticas calculadas com as pontuações do METEOR para 12918 sentenças do conjunto de testes.

([0-1]) que correspondem a 0,63% das ocorrências da distribuição, aproximadamente 8200 amostras dentro de tal intervalo, conforme exibe a Figura 4. Por último, analogamente a métrica da subseção anterior, fica evidente que a média de 0,1182 com desvio padrão de +/- 0,2249 está concentrada próxima do intervalo com as maiores ocorrências dentro da distribuição.

Figura 4 – Distribuição das pontuações do METEOR nas 12918 sentenças.



Os resultados obtidos foram muito baixos, piores que os resultados obtidos pelo BLEU, no entanto, não refletem a capacidade real de geração de títulos do modelo, que no geral foi muito boa e se mostrou cumprir com aquilo que foi postulado, mais adiante com a avaliação humana e com alguns exemplos de títulos gerados, ter-se-á uma ideia mais fiel da capacidade real do modelo.

3.6 Avaliação Humana - Medida-Locci

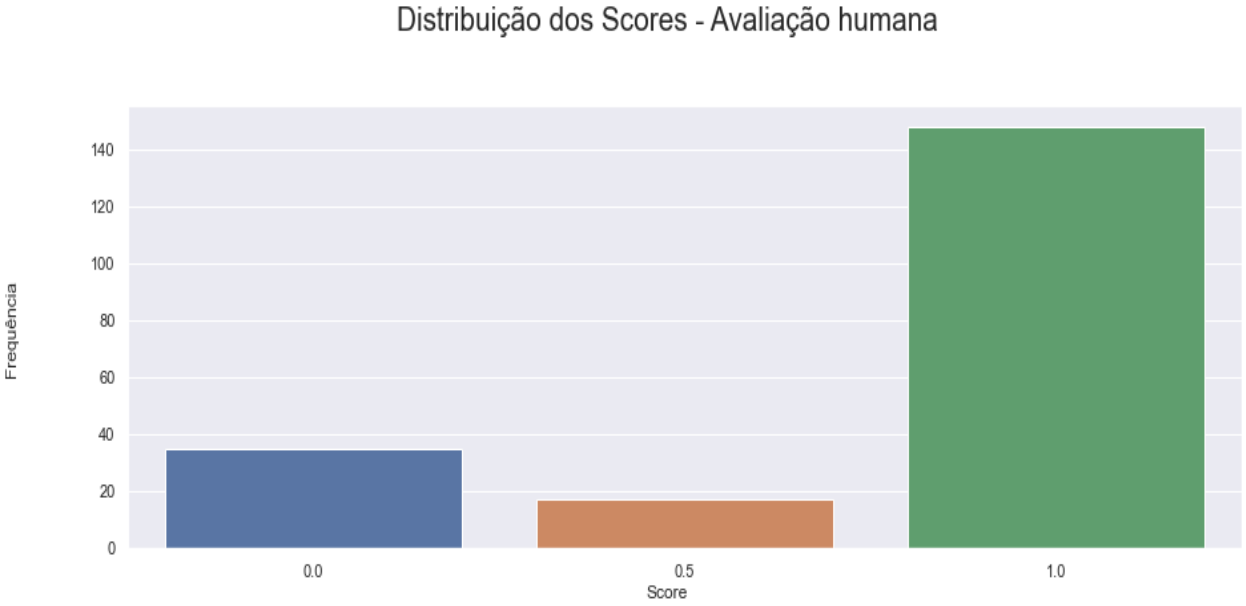
Para realizar a avaliação humana, um total de 200 sentenças da variável *text review* foram selecionadas ao acaso, utilizadas como entrada do modelo para prever os títulos, e a partir disso, construi-se um csv com as sentenças correspondentes ao *text review* e os títulos gerados para que o aluno Alexandre Locci avaliasse os títulos gerados com base no texto, atribuindo as pontuações sugeridas - 0 (ruim, incompreensível, totalmente não gramatical); 0,5 (mais ou menos passável) ou 1 (OK).

Estatísticas - Medida-Locci - 200 sentenças	
Minímo	0,0
Média	0,7825
Máximo	1,0
Desvio Padrão	0,3859

Tabela 7 – Estatísticas calculadas com as pontuações da Medida-Locci para 200 sentenças do conjunto de testes escolhidas ao acaso.

A Tabela 7 exibe as estatísticas dos dados da avaliação da Medida-Locci e a Figura 5 exibe a distribuição das pontuações de tal medida. A seguir uma discussão dos resultados é realizada.

Figura 5 – Distribuição das pontuações da Medida-Locci para 200 sentenças.



A menor pontuação foi 0 (zero), conforme exibe a Tabela 7, representando o segundo valor com mais ocorrências na distribuição - cerca de 35 sentenças, conforme a Figura 5 exibe. Assim, 17,5% das 200 sentenças selecionadas ao acaso, foram categorizadas como

ruim, incompreensível, totalmente não gramatical. Com menos ocorrências, a pontuação de 0,5 foi atribuída para um total de 17 sentenças, que corresponde a cerca de 8,5% da distribuição, pontuação com menor ocorrência.

O valor máximo - 1 - foi o que teve a maior ocorrência na distribuição, com 148 sentenças pontuadas, representa 74,0% da distribuição total. Isso mostra que a maioria dos títulos gerados estavam semanticamente coerentes com o texto de entrada, além disso, as previsões tem chance maior, quando não coerentes, de serem ruins ou incompreensíveis, em vez de ficarem no meio termo (mais ou menos passável).

A média de 0,7825 reflete de fato a quantidade de ocorrências da maior pontuação artribuída durante a análise dos títulos gerados. No entanto, o desvio padrão de +/- 0.3859 foi razoavelmente alto. Mesmo assim, os resultados apresentados pela Medida-Locci indicam que a capacidade do modelo em questão não é a mesma quando comparada com o resultados obtidos pelas métricas BLEU e METEOR, mesmo levando-se em consideração que apenas 200 amostras foram utilizadas na avaliação pela Medida-Locci contra 12918 amostras usadas na avaliação das medidas BLEU e METEOR.

3.7 Títulos gerados pelo modelo

A Figura 6 apresenta 8 amostras de títulos gerados pelo modelo *Encoder-Decoder BiLSTM*. Na sequência faremos uma breve analise.

Figura 6 – Amostras de títulos gerados pelo modelo.

```
[ 1 ] Text Review: produto em ordem e dentro do prazo de entrega. recomendo
Title Review: perfeito
Previsto: gostei do produto

[ 2 ] Text Review: atendeu as minhas expectativas e chegou antes do prazo recomendo
Title Review: gostei
Previsto: gostei muito do produto

[ 3 ] Text Review: recebi hoje. fui usar, que decepção, não funciona.
Title Review: decepção
Previsto: não funciona

[ 4 ] Text Review: tive problemas com a transportadora, porém o produto chegou ótimo e está sendo muito útil.
Title Review: adorei
Previsto: gostei muito do produto

[ 5 ] Text Review: o produto chegou dentro do prazo esperado, porém chegou cheio de imperfeições na superfície.
Title Review: produto com defeito
Previsto: gostei do produto

[ 6 ] Text Review: mochila feita de um tipo de napa , parece que irá rasgar fácil
Title Review: material ruim
Previsto: gostei muito do produto

[ 7 ] Text Review: quero devolução do meu dinheiro não recebi me devolvam meu dinheiro passou do prazo
Title Review: não foi entregue
Previsto: não recebi o produto

[ 8 ] Text Review: produto ruim. não atendeu às expectativas. tive que desmontar e utilizar somente um eixo pois f
icava folgado e balançando.
Title Review: péssima
Previsto: produto ruim
```

Os títulos 1, 2, 3 se mostram coerentes com o texto de entrada e com o título original, mesmo embora havendo um certo exagero no título 2, é possível observar que o título 1

carrega apenas o substantivo *produto* que também ocorre no texto. Por outro lado, o título 2 não carrega nenhuma palavra que ocorre no texto, no entanto, traz o verbo *gostei* que ocorre no título original. Em contrapartida o título 3 é formado por uma sub-frase que ocorre no texto - *não funciona*.

O título 4 também é coerente, mesmo havendo um pouco de negatividade no texto do usuário devido à problemas de entrega, o modelo conseguiu capturar a parte mais importante da frase, que capturou que o produto é bom visto que o usuário ao final afirmou que este tem sido muito útil.

Os títulos 5 e 6 são exemplos de casos em que o modelo errou na previsão. No título 5, uma possível hipótese para o erro cometido pelo modelo, pode ser que o mesmo tenha voltado a atenção para o inicio da frase, onde o usuário afirma que o produto chegou dentro do prazo, o que normalmente está fortemente relacionado a sentimentos positivos. Por outro lado, o título 6 não mostra nenhum indicio de sentimento positivo na frase, e mesmo assim, o título gerado foi o oposto do sentimento exposto pelo usuário. Em contrapartida os títulos 7 e 8 são exemplos de sentimentos negativos em que os títulos gerados foram semanticamente coerentes com o texto e o título original.

4 BERT

4.1 Estratégia de implementação e etapa adicional de preparação e preprocessamento dos dados

A etapa de preparação e preprocessamento adicional concernente ao modelo desta seção, foi influenciado pelas tentativas e fracassos obtidos com as estratégias de implementação realizadas. Para cumprir com o proposto, o modelo de referência *bert-base-portuguese-cased* foi selecionado como modelo pre-treinado e utilizado na realização de todos os experimentos.

A principio, a primeira estratégia foi a de se realizar o treinamento por refinamento como uma tarefa de classificação, e para tanto, utilizou-se favoravelmente da classe *TFBertForSequenceClassification*. Assim, a estratégia adotada para a preparação dos dados para alimentar o modelo foi a mesma sugerida no enunciado. O processo de treinamento e avaliação do modelo ocorreu normalmente, a estratégia adotada se mostrou promissora, até mesmo porque com está o treinamento do modelo era razoavelmente rápido.

Para realizar a implementação do *decoder*, no entanto, optou-se por usar favoravelmente do módulo *pipeline* da biblioteca *transformers*, de modo a se instanciar o objeto *unmasker*, onde o modelo *BERT*, o *tokenizer* e o método *fill-mask* são passados como parâmetros. Tal objeto possuí a característica de se passar uma sentença contendo o token MASK como parâmetro e retornar a palavra que prevista que substitui o token, daí o termo *unmasker*.

Porém, isso não foi possível de ocorrer, pois a classe *TFBertForSequenceClassification* não suporta fazer o desmascaramento do MASK em uma sentença. Essa situação, forçou a optar por outra estratégia que se seguiu por utilizar alternativamente a classe *TFBertForMaskedLM* que é específica para modelagem de linguagem mascarada e portanto tem suporte para tal operação.

A estratégia de preparação dos dados sugerida no enunciado se manteve, no entanto, sofreu uma pequena alteração, pois o modelo *TFBertForMaskedLM*, exige que as dimensões dos dados de entrada concordem com as dimensões dos dados de saída. Além disso, durante os experimentos foi evidenciado que o método *unmasker* do *pipeline* não é suscetível ao *fine-tuning*, ou seja, esse método não se adapta as novas amostras que são utilizadas no processo de transferência de aprendizado, assim, os resultados exibidos ao se utilizar esse módulo eram concernentes aos dados que foram utilizados no processo de pre-tre hípérparametrosinamento. Então, verificando a documentação do modelo, ficou evidente que para se realizar tal tarefa foi necessário implementar o *decoder* de maneira similar o que foi feito no modelo *Encoder-Decoder BiLSTM*, com método *predict*, que por sinal se mostrou efetivo da mesma forma, pois o modelo sabia exatamente como substituir o token MASK.

Os contratempos que surgiram influenciaram na preparação final dos dados antes de alimentarem o modelo, assim a preparação seguiu a estratégia sugerida no enunciado, porem, levando-se em consideração que os dados de entrada e saída precisam ter a mesma dimensão, a sentença de entrada precisou se repetir na sentença de saída, diferenciando-se da sentença de entrada com a adição da palavra em que se queria prever no lugar do token MASK. O exemplo a seguir ilustra como a transformação se deu:

Entrada: "[CLS] muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] [MASK] [SEP]"

Saída: "muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei"

Entrada: "[CLS] muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei [MASK] [SEP]"

Saída: "muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito"

Entrada: "[CLS] muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito [MASK] [SEP]"

Saída: “muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito do”

Entrada: “[CLS] muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito do [MASK] [SEP]”

Saída: “muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito do produto”

Entrada: “[CLS] muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito do produto [MASK] [SEP]”

Saída: “muito bom . câmera ótima . supera toda descrição do produto . eu super recomendo [SEP] gostei muito do produto [unused99]”

Tal transformação causou o efeito de aumentação de dados, assim, as 33093 amostras - 25% do conjunto completo - se transformaram em um total de 122685 amostras. Além disso, a quantidade máxima de palavras em ambas as variáveis de entrada - *text review* - e de saída - *title review* - sofreram alteração, para ambas as variáveis o tamanho máximo de palavras foi definido da seguinte forma:

Um máximo de 81 palavras por variável, onde dessas 81 palavras, 64 foram definidas previamente para a variável *text review*, 15 para o *title review* e mais 2 para os tokens [SEP] e [MASK], pois os tokens [CLS] e o token [SEP] que ocorre no final da sentença, são adicionados e contabilizados pelo *tokenizer* do BERT. Assim, salientamos que tais tokens não foram adicionados manualmente no processo de preparação dos dados, ou seja, apenas os tokens [SEP] [MASK], nesta ordem, que foram, no entanto os outros aparecem no exemplo acima para melhor ilustrar como as amostras ficam antes de serem convertidas para sequências numéricas.

A última etapa de preparação consistiu em se usar o *tokenizer BertTokenizer* do modelo de referência com os seguintes parâmetros configurados:

- is_split_into_words=True,
- padding=’max_length’,
- truncation=True,
- max_length=maxlen,

- pad_to_max_length=True,
- return_tensors='np',
- add_special_tokens=add_special_tokens

Devido ao *tokenizer* fazer parte de um modelo pré-treinado, ocorreu que não foi possível inserir um token particular de fim de sentença, como '[EOS]' por exemplo. Quando tal token era inserido manualmente, mesmo com auxílio de métodos do *tokenizer*, a *loss* calculada durante o processo de *fine-tuning* do modelo era afetada, e assim apenas exibia o valor *nan*. A solução para esse problema foi usar o token [unused99], configurado no posição de número 99 do dicionário com o vocabulário do *tokenizer*, que conta com 29794 palavras treinadas e foi utilizado para preparar ambos os dados do *text review* e do *title review*.

Por último, cabe destacar que a divisão dos dados em conjuntos de treino, validação e testes, foi realizada antes de aplicar as transformações mencionadas nesta seção com o intuito de evitar a sobreposição de dados entre os conjuntos.

4.2 Descrição das configurações e dos híper-parâmetros utilizados

A Tabelas 8 descreve o modelo BERT de escolha e as configurações e hiper-parâmetros que foram configurados na camada de *Input ids* e *Input masks ids*.

Modelo BERT e configurações das camadas de Input	
Modelo Bert	TFBertForMaskedLM
Modelo de Referência	BERTimbau
Input ids	81
Input maks ids	81

Tabela 8 – Modelo BERT de escolha e configurações camadas de Input ids e Input maks.

A classe *TFBertForMaskedLM*, exibida como modelo de escolha na tabela 8, é um modelo Bert com uma cabeça de modelagem de linguagem no topo, e conforme mencionado na seção anterior, foi selecionado para se utilizar favoravelmente do suporte do *unmasker* para se realizar as previsões em de palavras na posição em que o token MASK ocorre.

O *BERTimbau* foi escolhido como modelo de referência, também conhecido como *bert-base-portuguese-cased* pois é um modelo BERT pré-treinado para português brasileiro que atinge desempenhos de ponta em três tarefas de PNL downstream: Reconhecimento de Entidade Nomeada, Semelhança Textual de Frases e Reconhecendo Entailment Textual (vinculações linguísticas ocorrem quando se pode tirar conclusões necessárias de um uso específico de uma palavra, frase ou frase.) [Souza et al., 2020].

O tamanho da entrada para as camada de *Input ids* e *Input masks ids* exibido na Tabela 8, foi definido como 81, ou seja, a quantidade máxima de palavras por sentença, que foi definida para a variável de entrada do modelo - *text review* (64) - somada com a quantidade máxima de palavras por sentença, que foi definida para a variável de saída do modelo - *title review* (15) - somada com a quantidade de de tokens adicionados na etapa de preprocessamento manual - SEP e MASK (2).

No geral, as configurações e hiper-parâmetros se mantiveram os padrões do BERT e a implementação para a tarefa aqui proposta foi baseada no código da semana 10 disponibilizado em aula. Na seção a seguir a estratégia de inferência do modelo é descrita.

4.3 Processo de inferência do modelo

O processo de inferência foi implementado de modo a se receber uma ou mais sentenças sentença e gerar um título para ela(s). Para isso, as sentenças são inicialmente pre processadas e na sequência, para cada sentença, os tokens SEP e MASK são adicionados no final do texto.Então, os seguintes passos são realizados:

1. O texto é tokenizado e transformado em sequência númerica;

2. o indice da posição em que o token MASK ocorre é mantido em uma variável e utilizado para acessar a posição em que os títulos previstos são armazenados e recuperados de forma mais eficiente;

3. o *input ids* e o *attention mask* da sentença atual são usados para se prever o título correspondente com método *predict* do modelo;

4. a saída será uma matriz de probabilidades, e a maior probabilidade em cada linha da matriz representa a próxima palavra em sua respectiva posição na sentença e é selecionada com *argmax*; (O modelo sempre preve as mesmas palavras que formam a sentença de entrada, com a exceção que na posilção do MASK é que temos o título previsto que de fato queremos);

5. a sequência prevista é convertida para tokens e os tokens para string; O token MASK é adicionado ao final da string prevista para continuar a previsão até que está termine ao encontrar o token de fim de sentença ou um tamanho máximo for atingido;

6. o indice do token MASK é utilizado para recuperar a palavra prevista pelo modelo. A palavra é armazenada em uma lista de palavras que ao final formarão o título previsto pelo modelo;

7. os passos de 3 a 6 são repetidos até que o token de fim de sentença ocorra ou que um tamanho máximo seja atingido;

8. por último os títulos previstos são retornados para o mensurar as pontuações com as métricas propostas no enunciado.

4.4 Treinamento e Validação

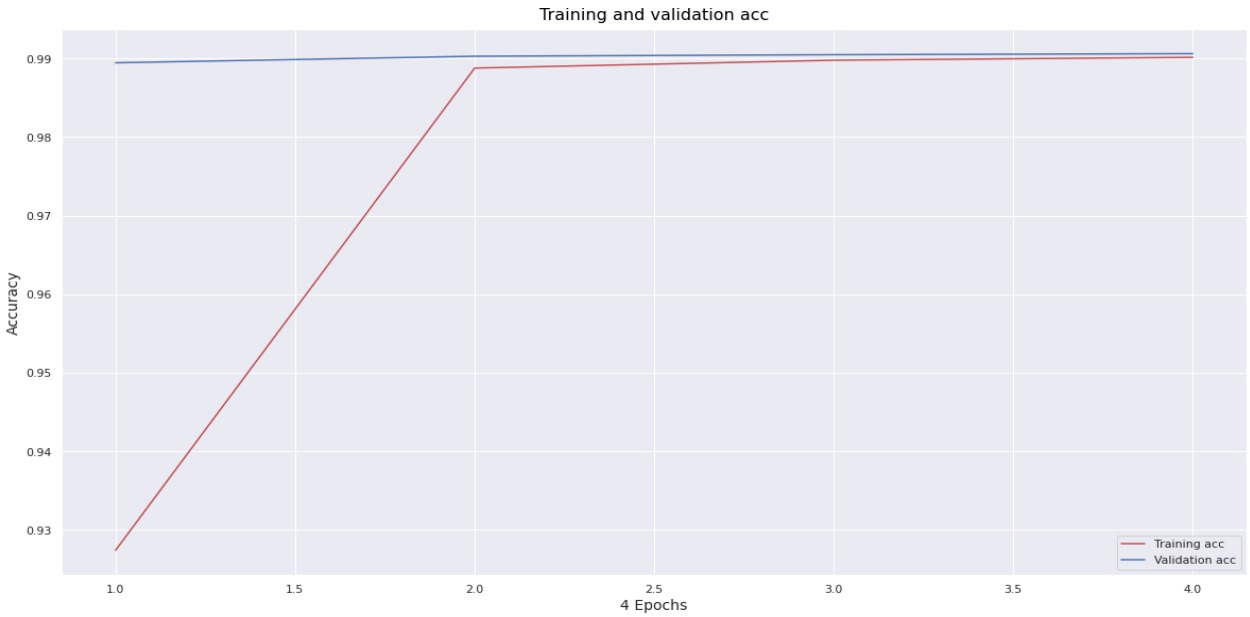
A função *loss* utilizada foi a *Sparse Categorical Crossentropy*, e o algoritmo Adam foi utilizado como função de otimização. A taxa de aprendizado foi configurada com 1e-5 e ε como 1e-07, os outros parâmetros do Adam não foram alterados, manteu-se os padrões do Keras que segue o artigo original. O tamanho de cada lote (*batch size*) utilizado durante a etapa de treinamento foi definido como 64 amostras por lote. Para o treinamento um total de 4 épocas foram estipuladas.

Para monitorar o desempenho do modelo, o *callback Early Stopping* foi configurado para monitorar o desempenho da *loss* com base no conjunto de validação. Especificamente, o *Early Stopping* foi configurado para interromper o treinamento do modelo quando não houver melhora de desempenho dentro de 2 épocas de treinamento. Por último, os dados utilizados para treinar, validar e testar o modelo, foram particionados na proporção 80% para o conjunto de treinamento; 10% para o conjunto de validação e 10% para o conjunto de testes. Assim, o modelo contou com um total de 98155 amostras no conjunto de treino; 12204 amostras no conjunto de validação e 12326 amostras no conjunto de teses (lembrando que os dados foram particionados antes do processo de preparação e tokenização, por isso existe essa pequena discrepância entre os conjuntos de teste e validação).

A Figura 7 exibe o desempenho do modelo durante o processo de treinamento e validação do modelo. Durante tal etapa, a melhor acurácia média obtida no conjunto de treinamento foi 99,01% durante a época de número de 4. A melhor acurácia média no conjunto de validação foi de 99,06% durante a época de número 4. O treinamento não sofreu overfitting, no entanto, poucas épocas foram estipuladas pois o treinamento de cada época neste modelo demorou em média 30 minutos.

Após está etapa o modelo foi testado no conjunto de testes que obteve uma acurácia média de 99,06%. Mas, como essa medida não é confiável para a tarefa de geração de textos, outras métricas e metodologias foram utilizadas para avaliar o modelo no conjunto de testes, que são descritos na seção a seguir.

Figura 7 – Curvas de acurácia do modelo BERT nos conjuntos de treino e validação.



4.5 Testes e Resultados

Para se realizar os testes, de maneira similar ao modelo *Encoder-Decoder BiLSTM*, além da acurácia média, duas métricas adicionais foram consideradas. As métricas escolhidas foram o BLEU, e o METEOR, que foram utilizadas com auxílido da biblioteca de processamento de linguagem natural NLTK. Também se realizou uma avaliação do texto gerado com auxilio de um ser humano. Tal avaliação foi realizada considerando-se 200 amostras de texto geradas pelo modelo e selecionadas ao acaso. O aluno e colega de pós-graduação Alexandre Locci foi quem realizou tal avaliação, atribuindo as pontuações 0, 0,5 e 1,0 conforme solicitado para este trabalho, assim doravante denominaremos a avaliação humana de medida-Locci.

O número de amostras usadas para avaliar os títulos gerados foram apenas 3268, pois, para as transformações que ocorreram no conjunto de dados com o efeito de aumentação foram apenas necessárias para a tarefa de treinamento e validação do modelo, para calcular as pontuações isso não foi necessário, pois o processo de inferência já utiliza uma estratégia de autorregressão para gerar o título a partir das previsões anteriores e por isso o conjunto de testes se encontrou reduzido nesta fase.

4.5.1 BLEU

O BLEU implementado no NLTK testa de 1 até 4-grams as sobreposições do texto gerado em comparação com o original. Além disso, para se computar as pontuações foi necessário converter as 3268 sequencias do conjunto de texto para o formato de texto (string). Isso ocasionou em um tempo de 7 minutos para se calcular as pontuações. A tabela 9 exibe as

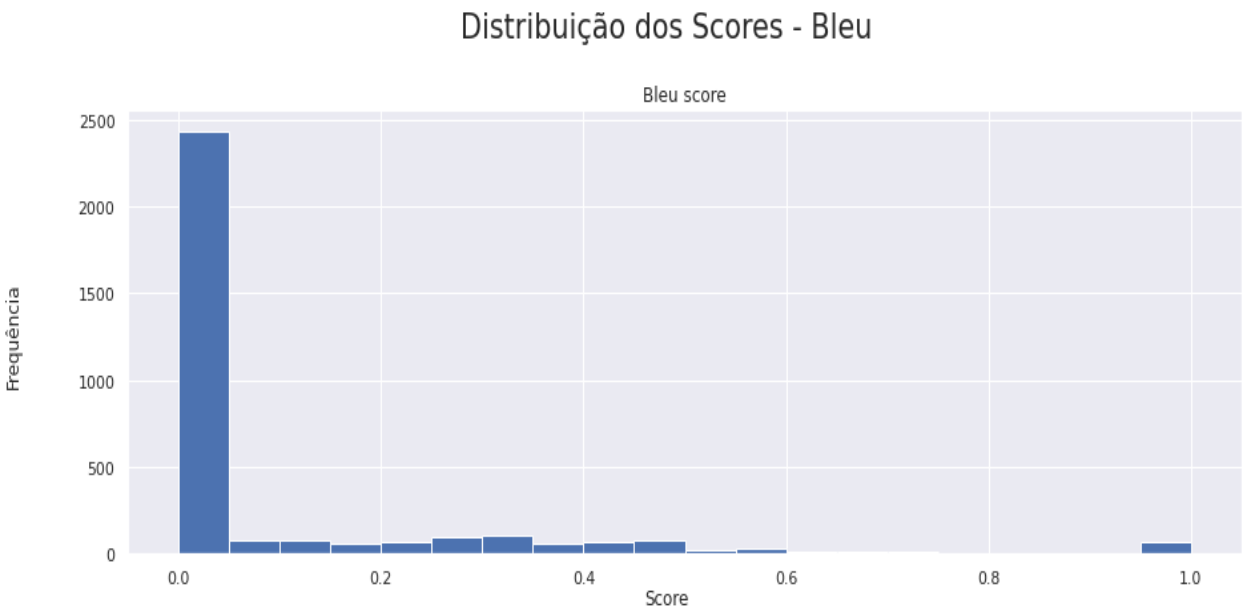
estatísticas calculadas com as pontuações do BLEU nas amostras testadas.

A melhor pontuação foi o valor máximo de 1, conforme exibido na tabela 9, o que de acordo com a Figura 8 significa que o modelo previu aproximadamente 100 títulos idênticos aos originais - cerca de 3,06% da distribuição. Por outro lado, a menor pontuação prevista - 0 (zero) - está dentro do grupo com maior frequência de pontuações, aproximadamente 2450 amostras dentro de tal faixa, conforme se pode observar na Figura 8, que se encontra dentro de um intervalo de pontuações entre 0 e 0,1, aproximadamente, o que representa 75% das pontuações calculadas. Essa observação fica um pouco mais evidente ao se observar que a média das pontuações foi de 0,0966 com desvio padrão de +/- 0,2065, razoavelmente distante da média. Embora o conjunto de dados seja menor aqui, os resultados foram bem semelhantes ao do modelo *Encoder-Decoder BiLSTM*.

Estatísticas - BLEU - 3268 sentenças	
Minímo	0,0
Média	0,0966
Máximo	1,0
Desvio Padrão	0,2065

Tabela 9 – Estatísticas calculadas com as pontuações do BLEU para 3268 sentenças do conjunto de testes.

Figura 8 – Distribuição das pontuações do BLEU nas 3268 sentenças.



Os resultados obtidos foram muito baixos, no entanto, não refletem a capacidade real de geração de títulos do modelo, que no geral foi muito boa e se mostrou cumprir com aquilo que foi postulado, mais adiante com a avaliação humana e com alguns exemplos de títulos gerados, ter-se-á uma ideia mais fiel da capacidade real do modelo.

4.5.2 METEOR

O METEOR levou um tempo menor que BLEU, apenas 2 minutos para calcular as pontuações para todas as sentenças do conjunto de testes. A Tabela 10 a seguir apresenta os resultados obtidos calculando-se as pontuações do METEOR com auxílio da biblioteca NLTK no conjunto de testes.

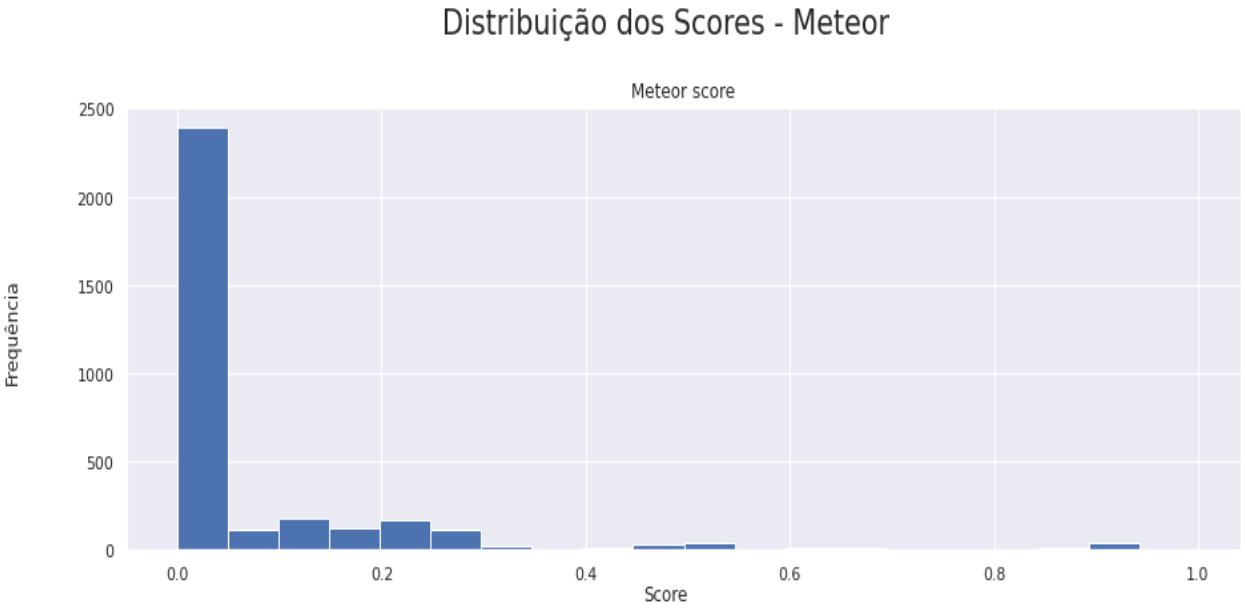
Estatísticas - METEOR - 3268 sentenças	
Minímo	0,0
Média	0,0698
Máximo	0,9922
Desvio Padrão	0,1583

Tabela 10 – Estatísticas calculadas com as pontuações do METEOR para 3268 sentenças do conjunto de testes.

A maior pontuação foi de 0.9922, cerca de 2%, o que representa aproximadamente 50 amostras da distribuição, conforme se pode observar na Figura 9. Em contrapartida, a menor pontuação prevista foi 0 (zero), ou seja, de maneira semelhante ao BLEU, tem se a ocorrência títulos previstos em que as palavras formadoras da sentença diferem totalmente daquelas que formam o título original. Tal pontuação mínima se encontra dentro de uma faixa de valores ([0-1]) que correspondem a 69% das ocorrências da distribuição, aproximadamente 2250 amostras dentro de tal intervalo, conforme exibe a Figura 9. Por último, analogamente a métrica da subseção anterior, fica evidente que a média de 0,0698 com desvio padrão de +/- 0,1583, menor que no BLEU, está concentrada próxima do intervalo com as maiores ocorrências dentro da distribuição.

Os resultados obtidos foram muito baixos, piores que os resultados obtidos pelo BLEU, no entanto, não refletem a capacidade real de geração de títulos do modelo, que no geral foi muito boa e se mostrou cumprir com aquilo que foi postulado, mais adiante com a avaliação humana e com alguns exemplos de títulos gerados, ter-se-á uma ideia mais fiel da capacidade real do modelo.

Figura 9 – Distribuição das pontuações do METEOR nas 3268 sentenças.



4.6 Avaliação Humana - Medida-Locci

Para realizar a avaliação humana, um total de 200 sentenças da variável *text review* foram selecionadas ao acaso, utilizadas como entrada do modelo para prever os títulos, e a partir disso, construi-se um csv com as sentenças correspondentes ao *text review* e os títulos gerados para que o aluno Alexandre Locci avaliasse os títulos gerados com base no texto, atribuindo as pontuações sugeridas - 0 (ruim, incompreensível, totalmente não gramatical); 0,5 (mais ou menos passável) ou 1 (OK).

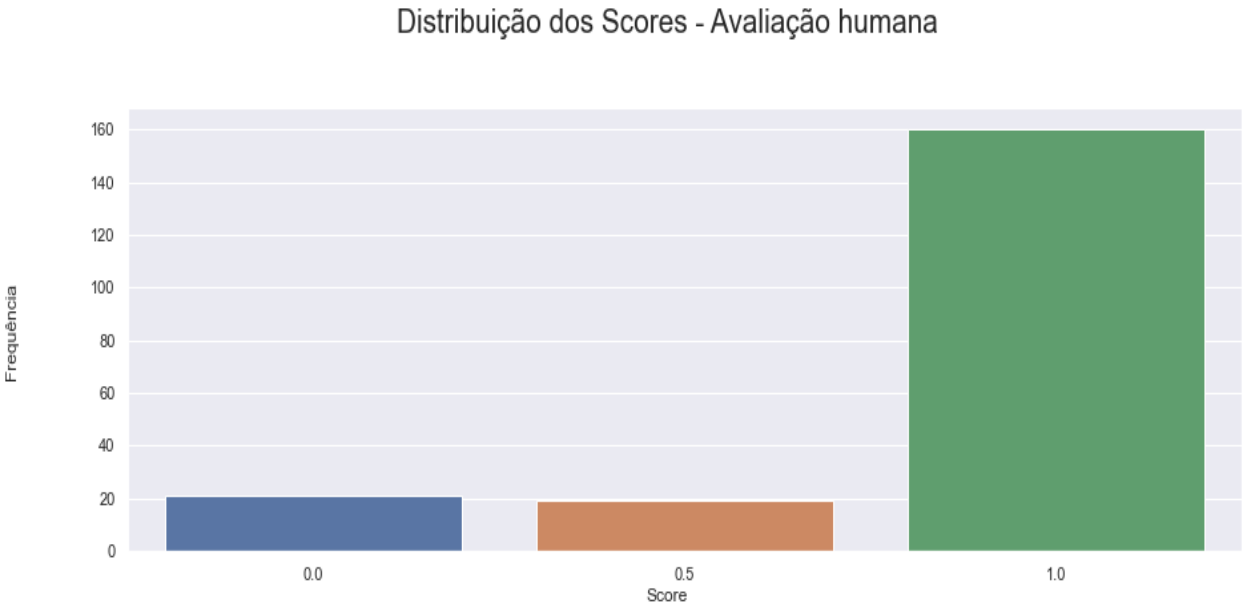
Estatísticas - Medida-Locci - 200 sentenças	
Minímo	0,0
Média	0,8475
Máximo	1,0
Desvio Padrão	0.3248

Tabela 11 – Estatísticas calculadas com as pontuações da Medida-Locci para 200 sentenças do conjunto de testes escolhidas ao acaso.

A Tabela 11 exibe as estatísticas dos dados da avaliação da Medida-Locci e a Figura 10 exibe a distribuição das pontuações de tal medida. A seguir uma discussão dos resultados é realizada.

A menor pontuação foi 0 (zero), conforme exibe a Tabela 11, representando o segundo valor com mais ocorrências na distribuição - cerca de 21 sentenças, conforme a Figura 10 exibe. Assim, 10,5% das 200 sentenças selecionadas ao acaso, foram categorizadas como ruim, incompreensível, totalmente não gramatical. Com menos ocorrências, a pontuação

Figura 10 – Distribuição das pontuações da Medida-Locci para 200 sentenças.



de 0,5 foi atribuída para um total de 19 sentenças, que corresponde a cerca de 9,5% da distribuição, pontuação com menor ocorrência.

O valor máximo - 1 - foi o que teve a maior ocorrência na distribuição, com 160 sentenças pontuadas, representa 80,0% da distribuição total. Isso mostra que a maioria dos títulos gerados estavam semanticamente coerentes com o texto de entrada, além disso, as previsões tem chance maior, quando não coerentes, de serem ruins ou incompreensíveis, em vez de ficarem no meio termo (mais ou menos passável).

A média de 0,8475 reflete de fato a quantidade de ocorrências da maior pontuação artribuída durante a análise dos títulos gerados. No entanto, o desvio padrão de +/- 0.3258 foi razoavelmente alto, um pouco menos que no *Encoder-Decoder BiLSTM*. Mesmo assim, os resultados apresentados pela Medida-Locci indicam que a capacidade do modelo em questão não é a mesma quando comparada com o resultados obtidos pelas métricas BLEU e METEOR, mesmo levando-se em consideração que apenas 200 amostras foram utilizadas na avaliação pela Medida-Locci contra 3268 amostras usadas na avaliação das medidas BLEU e METEOR.

4.7 Títulos gerados pelo modelo

A Figura 11 apresenta 7 amostras de títulos gerados pelo modelo *BERT*. Na sequência faremos uma breve analise.

Todos os títulos que foram selecionados, se mostram coerentes com o texto de entrada e com o título original, obviamente existem amostras que são incoerentes, mas não foi o

Figura 11 – Amostras de títulos gerados pelo modelo.

```
[ 1 ] Text Review: ótimo produto e a entrega muito antes do prazo estipulado.
Title Review: ótimo produto
Previsto: ótimo produto

[ 2 ] Text Review: excelente produto, chegando antes do prazo previsto. superindico
Title Review: entrega antes do prazo previsto
Previsto: excelente

[ 3 ] Text Review: não posso avaliar um produto que comprei a mais de trinta dias e ainda não foi entregue
Title Review: onde está o produto?
Previsto: não posso avaliar

[ 4 ] Text Review: algumas peças vieram riscadas, e vieram faltando porcas. fora o atraso.
Title Review: peças vieram riscadas e faltando peças
Previsto: produto com defeito

[ 5 ] Text Review: acabou as chamadas indesejaveis, com pertubação de a ligações por dia.
Title Review: gostei do produto
Previsto: péssimo

[ 6 ] Text Review: não recebi ainda não recebi ainda não recebi ainda
Title Review: não recebi ainda
Previsto: não recebi ainda

[ 7 ] Text Review: o produto é de boa qualidade. fui muito bem atendido.
Title Review: gostei do produto
Previsto: gos gos muito bom
```

caso nas amostras selecionadas ao acaso para exibição nesta seção. De fato, o título 7 gerou algumas palavras sem sentido no inicio da sentença, mesmo assim, acertou. Podemos ainda observar que o título gerado tem base totalmente no *text review* e muitas vezes acabou até mesmo repetindo sub-frases que ocorreram neste, conforme ocorreu nas sentenças 1, 2, 3 e 6.

Em relação ao *Encoder-Decoder BiLSTM*, os títulos com sentimento negativo foram igualmente coerentes como foram aqueles com sentimento positivo e no geral as previsões foram ótimas.

5 Comparação entre os modelos, discussão dos resultados e escolha do melhor gerador de títulos

A Tabela 12 exibe um resumo com a média dos resultados obtidos pelos modelos propostos. Os resultados são comparados, discutidos e um dos modelos é eleito como o melhor gerador de títulos.

Resumo das estatísticas dos modelos Encoder-Decoder BiLSTM e BERT				
	Acurácia	BLEU	METEOR	Medida-Locci
Encoder-Decoder BiLSTM	82,50%	**0,1459 +/- 0,2578**	**0,1182 +/- 0,2249**	0,7825 +/- 0,3859
BERT	**99,06%**	0,0966 +/- 0,2065	0,0698 +/- 0,1583	**0,8475 +/- 0,3248**

Tabela 12 – Resumo das estatísticas calculadas com os modelos Encoder-Decoder BiLSTM e BERT.

O modelo *BERT* teve uma acurácia superior ao modelo *Encoder-Decoder BiLSTM* conforme destacado na Tabela 12, 99,06% contra 82,50%, cerca de 16,6% superior. No entanto,

nos atentaremos mais as outras métricas visto que a acurácia não é um bom indicativo de avaliação para o problema de tradução / sumarização de textos.

Assim, podemos observar, de acordo com a Tabela 12, que o modelo *Encoder-Decoder BiLSTM* superou o modelo *BERT* nas medidas BLEU e METEOR, com médias de 0,1459 +/- 0,2578 e 0,1182 +/- 0,2249 contra as médias de 0,0966 +/- 0,2065 contra 0,0698 +/- 0,1583, respectivamente. Em contrapartida, o *BERT* superou *Encoder-Decoder BiLSTM* nas avaliação humana com a média de 0,8475 +/- 0,3248 contra 0,7825 +/- 0,3859, respectivamente.

O modelo *Encoder-Decoder BiLSTM* foi superior na maior parte das métricas utilizadas, no entanto, considerando-se que o conjunto de dados utilizado no *BERT* para avaliar foi um conjunto reduzido daquele usado no *Encoder-Decoder BiLSTM*, pode-se considerar que os resultados foram similares, além disso, é importante se observar que o desvio padrão dos resultados no *Encoder-Decoder BiLSTM* foram maiores do que no *BERT* - 2,54 vezes maior. Por último, a média da pontuação da avaliação humana no *BERT* foi superior a média da avaliação humana no *Encoder-Decoder BiLSTM*, em cerca de 7% de diferença, ou seja, 0,8475 +/- 0,3248 contra 0,7825 +/- 0,3859 respectivamente. Deste modo, levando-se em consideração a importância da avaliação humana em relação as outras métricas, mesmo com um conjunto de dados limitados à 200 sentenças, e, os desvios padrão que foram menores no *BERT* em relação ao *Encoder-Decoder BiLSTM*, nos dão indícios que o modelo *BERT* teve um desempenho estatísticamente superior ao modelo *Encoder-Decoder BiLSTM*, e portanto o declaramos como o melhor gerador de títulos deste experimento.

6 Conclusões

A tarefa de sumarização / tradução de textos é uma tarefa complexa e exige métricas adequadas para se poder mensurar de maneira mais confiável o desempenho real de um modelo de linguagem treinado para a tal tarefa. Durante o decorrer deste exercício programa diversos desafios surgiram, desde o primeiro momento em que se definiu em abordar o problema proposto como uma tarefa de sumarização de textos, até o final em que se adotaram estratégias peculiares para implementar um decoder usando o BERT.

A implementação do modelo *Encoder-Decoder BiLSTM* foi mais natural, embora com diversos detalhes para se levar em consideração, principalmente ao se considerar cada parte que o constitui - camadas decoder, encoder, atenção, etc. Por outro lado, com o BERT diversos experimentos e tentativas precisaram ser realizadas, pois a documentação não contribuiu por diversas vezes quando foi necessário se aprofundar em alguma particularidade de alguma classe e/ou módulo provido pela api, o que acabou levando um tempo maior para se concretizar durante o desenvolvimento deste trabalho.

Os resultados obtidos foram surpreendentes, mesmo embora as baixas pontuações nas métricas propostas, ficou evidente ao se realizar uma analise minuciosa na saída gerada pelo modelo, que os títulos gerados eram em sua grande maioria, semanticamente coerentes com o texto do usuário.

Mesmo com os resultados ruins, ainda assim, os resultados apenas fornecem uma vaga ideia de como um modelo treinado para a tarefa de geração de títulos se comportaria, pois, os resultados apresentados não podem ser considerados sólidos, ou seja, estatísticamente confiáveis. Afirmamos isso, pois foram realizados apenas um experimento de treinamento, avaliação e testes com os modelos completos. Isso se deu devido ao tempo demasiado que os modelos levavam para realizar o treinamento - 1,65 minutos no *Encoder-Decoder BiLSTM* para cada época de treinamento; 36 minutos para computar o BLEU e o METEOR durante a fase de testes; e 38 minutos no *BERT* cada época de treinamento; 7 minutos para computar o BLEU e 2 minutos para computar o METEOR - além disso os recursos do *google colaboratory* possuem limite de uso, o que impediu a realização de mais testes a fim de fornecer resultados mais consistentes.

Para trabalhos futuros propomos que métricas mais apropriadas para esse tipo de tarefa sejam usadas, como a perplexidade por exemplo. Além disso, se possível, utilizar uma quantidade maior de indivíduos para avaliarem as sentenças geradas, assim uma quantidade maior de sentenças também podem ser avaliadas. Um outro possível método de avaliação que funcionaria bem para o problema aqui abordado em especifico, seria a utilização de uma modelo de classificação de sentimentos para classificar os títulos. Por último e o mais essencial é importante e fundamental se ter disponível equipamentos com GPU's sem limite de uso para se ter a liberdade de realizar uma gama maior de experimentos.

Referências

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

[Callison-Burch et al., 2006] Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-

evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

[Ganegedara, 2020] Ganegedara, T. (2020). *Keras Attention Layer Implementation.* Disponível em: `https://github.com/thushv89/attention_keras/`. Acesso em: 11 de Dezembro de 2020.

[Souza et al., 2020] Souza, F., Nogueira, R., and Lotufo, R. (2020). BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear).*