DESENVOLVIMENTO DE UMA APLICAÇÃO DE TRADUÇÃO DA LÍNGUA TUPI PARA A LÍNGUA PORTUGUESA, POR MEIO DE TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL (NLP)

André Alves da Cruz[1], Yuri de Almeida Malheiros Barbosa[1]

[1]Departamento Ciências Exatas (DCX) – Universidade Federal da Paraíba(UFPB

Rio Tinto – PB, 58297-000
{andre.alves, yuri}@dcx.ufpb.br

Abstract. *As time passed, the Tupi language ceased to be spoken as the main language by the indigenous tribes, coming to be considered an extinct language. With the rapid advancement of technology, many techniques have emerged that made it possible to perform tasks that, just over 10 years ago, were tasks that were extremely determined to be performed. From these scenarios, the present work tries to use the natural language processing method, for the development of a translation application that provides translations of words and phrases from the Tupi language into Portuguese language.*

Resumo. *Ao passar do tempo, a língua Tupi, deixou de ser falada como a principal língua pelas tribos indígenas, chegando a ser considerada uma língua extinta. Com o rápido avanço da tecnologia, surgiram muitas técnicas que proporcionaram realizar tarefas que a pouco mais de 10 anos atrás, eram consideradas tarefas extremamente difíceis de serem realizadas. A partir destes cenários, o presente trabalho tentar utilizar técnicas de processamento de linguagens natural, para o desenvolvimento de uma aplicação de tradução que forneça traduções de palavras e frases da língua Tupi para a íngua Portuguesa.*

1.0 INTRODUÇÃO

O Brasil é um dos país mais ricos culturalmente, contendo não só culturas típicas brasileiras, mas também de diversos outros países. De acordo com Vivian Campos (2017), a formação da cultura brasileira se deu a partir da miscigenação de vários grupos étnicos, porém foram dos portugueses de quem herdamos a herança cultural fundamental, incluindo a língua portuguesa.

[1]"Trabalho de conclusão de curso, sob orientação do professor Yuri de Almeida Malheiros Barbosa submetido ao Curso de Bacharelado em Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO."

Segundo Isabel Alves (2019) no início da colonização do Brasil em 1500 por Portugal, existiam diversas línguas indígenas que eram faladas pelos nativos. Os mesmos faziam parte de um grupo indígena conhecido como tupi, nome dado também para a língua que as tribos falavam.

1.1 Problema

De acordo com Ana Paula (2019) no ano de 1757 uma Provisão Real proibiu a utilização da língua Tupi, não apenas pelos portugueses, mas também pelos povos nativos da região, na época o Português já suplantava esta língua, ficando ele, o Português, com o título de idioma oficial.

Aryon Dall'Igna (2007) explica que após a proibição da língua Tupi, as tribos indígenas continuaram a falar a língua, pois era sua língua nativa e sentiam dificuldade em aprender a língua portuguesa, o que fez com que os portugueses perseguissem e punisse os indígenas que ainda falassem a sua língua nativa. Aryon Dall'Igna também explica que muitas das tribos se isolaram em locais de difícil acesso, e com o passar do tempo cada tribo passou a ter a sua própria variação da língua Tupi, todas advindas de uma mesma origem.

1.2 Motivação

Segundo John M. Monteiro (2001) ao decorrer do tempo, o Tupi antigo deixou de ser falado como a principal língua por tribos indígenas, e chegou a ser considerado uma língua indígena extinta, pois não havia registros de nenhuma tribo indígena que ainda falasse tal língua.

Com o avanço da tecnologia, surgiram técnica capazes de recuperar imagens antigas, prever acontecimentos futuros ou até mesmo de realizar traduções de texto entre duas línguas distintas. Helena de Medeiros (2017) apresenta algumas técnicas de processamento de linguagem natural que são capazes de realizar traduções automática de texto utilizando técnicas de aprendizagem de máquina.

A partir deste cenário surgiu a ideia de se utilizar técnicas de processamento de linguagem natural, capazes realizar traduções da língua Tupi para a língua Portuguesa. Desta forma vindo a facilitar o acesso das pessoas a língua Tupi, ou até mesmo possibilitando com que as pessoas aprenderam tal língua.

Para tal objetivo ser alcançado, se fez necessário a realização de algumas tarefas de suma importância. Como a digitalização de materiais didáticas da língua Tupi, pois não havia material digitalizado que poderia ser utilizar como fonte de dados.

Outro passo importante para se atingir o objetivo, foi a criação de um *dataset* da língua Tupi para a língua Portuguesa, criado a partir de frases coletadas dos livros digitalizados. Em seguida se deu início ao o desenvolvimento do algoritmo de tradução, utilizando técnicas de processamento de linguagem natural, o mesmo podendo ser utilizado para o desenvolvimento de uma aplicação de tradução.

2. FUNDAMENTAÇÃO TEÓRICA

A seguir serão apresentados os assuntos que deram a base para o desenvolvimento deste trabalho.

2.1 Processamento de Linguagem Natural

Segundo Luís Moniz (2003) o processamento de Linguagem Natural, tem como objetivo estudar as limitações de uma máquina em entender a forma como os seres humanos se comunicam. Além de fornecer aos computadores a capacidade de entender e der gerar textos, sendo possível desenvolver soluções para que os computadores possam entender a forma como os humanos se comunicam e possam replicar a mesma.

Neste trabalho foi utilizado técnicas de processamento de linguagem natural juntamente com algoritmos de aprendizagem de máquina, que utilizam redes neurais artificiais para processar as informações.

2.2 Redes Neurais Artificiais

A primeira rede neural foi criada por Warren McCulloch e Walter em 1943. Eles escreveram um artigo chamado “A logical Calculus of the Ideas Immanent in Nervous Activity” no mesmo ano, no qual apresentaram os conceitos iniciais de como uma rede neural artificial funcionária, em seguida desenvolveram uma rede neural utilizando circuitos elétricos para demonstrar o seu funcionamento.

Os algoritmos de redes neurais são capazes de reconhecer padrões escondidos e fazer correlações com os dados brutos existentes, além de agrupá-los, classificá-los e até mesmo melhorá-los.

As redes neurais mais simples, são chamadas de perceptron, o perceptron é um algoritmo simples que é destinado a realizar classificações binárias. Ele consegue prever se uma determinada entrada pertence a uma determinada categoria ou não.

2.2.1 Redes Neurais Recorrentes

Segundo Rodrigo e Luan (2010) as redes neurais recorrentes (*RNN*) são formadas por um conjunto de redes neurais artificiais, que são utilizadas para realizar o processamento de sequências de dados, como som, voz ou sequências de texto por exemplo.

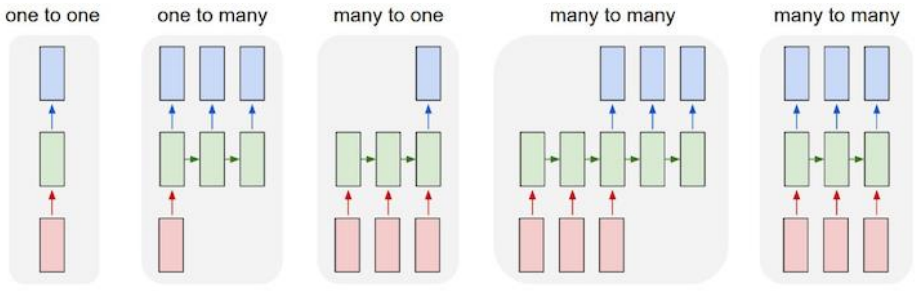


Figura 1: Arquitetura de entrada e saída das RNN. Fonte: Attention is all you need.

A figura 1 mostra uma visão alto nível de uma rede *RNN*, diferentemente de como um perceptron funciona, uma rede *RNN* pode variar na quantidade de entradas e saídas da rede, podendo haver apenas uma única entrada com uma única saída ou várias entradas com várias saídas.

As *RNNs* se diferenciam das demais redes neurais, porque incluem um loop de feedback pelo qual a saída da etapa T-1 é alimentada de volta à rede, para assim influenciar o resultado da etapa T. Esse processo de feedback se repete para cada uma das etapas subsequentes, gerando algo similar a uma memória interna dentro do algoritmo.

2.2.2 O problema das RNNs

As *RNNs* têm um problema chamado de *exploding vanishing gradientes*, explicado por Gustavo Bennemann (2018) em seu artigo sobre Redes Neurais Recorrentes. Tal problema acontece na maioria das vezes quando temos uma sequência muito grande de dados, onde para se processar as saídas da rede é necessário analisar cada tempo anterior. Algo que dependendo do tamanho da sequência de dados, exige muito processamento tornando a rede muito lenta ou até inviável.

2.3 Redes Neurais Sequence to Sequence

Para amenizar o problema das *RNNs* clássicas, surgiu a rede Sequence to Sequence (*Seq2Seq*), que é uma *auto-encoder* recorrente formada por duas redes recorrentes. Tal Rede neural foi apresentada inicialmente por Kyunghyun (2014). Esta nova rede neural seria capaz atribuir pesos de importância para cada palavra repassada como entrada para a rede

Uma rede pode por exemplo, aprende a comprimir textos em Tupi e a outra rede aprender a descomprimir em Português, realizando o mapeamento de sequências com sequências. As redes *Seq2Seq* contam com um conceito de atenção, para gerar a saída são observados os parâmetros anteriores e o grau de importância de cada parâmetro dentro do contexto no qual está inserido.

No entanto, mesmo as sedes *Seq2Seq* utilizando um mecanismo de atenção, quando passamos um texto muito extenso, ainda existe o mesmo problema das redes *RNNs*, principalmente para as últimas palavras inseridas, pois precisam analisar todas as palavras anteriores, mesmo com cada palavra tendo pesos diferentes para o processamento.

2.4 Arquitetura Transformer

A arquitetura *Transformer* foi criada por uma equipe de engenheiros do google e foi apresentada no artigo "*Attention Is All You Need*" (2017), neste artigo os engenheiros apresentaram uma nova arquitetura para solucionar o problema da alta dependência das redes neurais *Seq2Seq*.

A arquitetura *Transformer* funciona de forma muito parecida com a arquitetura *Seq2Seq*, existe uma estrutura de *encoder* responsável por ler as entradas e uma estrutura de *decoder* responsável por fazer as previsões das palavras.

Uma das principais diferenças entre as duas arquiteturas, é que na *Seq2Seq* alimentamos a rede palavra por palavra, que gera uma dependência muito grande, já na arquitetura *Transformer*, passamos frases inteiras para a rede, desta forma a arquitetura analisa o contexto de uma frase por inteiro sem gerar uma grande dependência.

Nesta arquitetura também temos dois mecanismos de atenção, responsáveis por tarefas distintas. O primeiro deles é o *Scaled Dot-Product Attention*, que faz a recomposição de uma sequência para saber como cada elemento está relacionado com os outros da mesma frase. Além do *Scaled Dot-Product Attention* temos o *Look-ahead mask Attention* que relaciona as os pares de frases passados como entrada, os dois mecanismos de atenção trabalham em conjunto durante o processo treinamento para prever as saídas da rede.

3. METODOLOGIA

Para a construção deste trabalho foram realizadas algumas etapas essenciais, que serão detalhadas nas subseções abaixo.

3.1 Dataset Tupi-Português

O *dataset* Tupi-Português foi criado tendo-se como base outros *datasets* públicos disponíveis na web, como os descritos por Koehn Philipp (2011) que reuniu 20 *datasets* e os disponibilizou publicamente em *statmt.org/europarl*. Neste artigo será utilizado o *dataset* Português-Inglês como base para fazer realizar os testes durante o desenvolvimento do algoritmo.

Para o desenvolvimento deste trabalho, foi necessário a criação de um *dataset* contendo pares de frases da língua Tupi para a língua Portuguesa, isto foi necessário pois os algoritmos de tradução automática que serão implementados, são treinados partir de pares de frases de uma língua para outra. Para cada frase em uma língua específica temos a sua frase correspondente na outra língua, como não existia *datasets* da língua Tupi para a língua Portuguesa, foi necessária a criação desta base de dados.

Para a construção do *dataset*, foi necessário fazer a coleta de materiais da língua Tupi que continham pares de frases ou palavras Tupi-Português ou vice-versa. Também foi necessário fazer a digitalização de alguns livros para ser possível fazer a extração dos pares de frases dos mesmos.

O principal livro usado como material para a criação da base de dados, foi o livro "Método Moderno de Tupi Antigo Segunda Edição" (EDUARDO NAVARRO, 2013), juntamente com o "Dicionário Tupi Antigo: a Língua Indígena Clássica do Brasil" (EDUARDO NAVARRO, 2013). Estas foram as referências mais recentes para a criação do

dataset, que até a data de escrita deste trabalho, contém 2 mil pares de frases Tupi-Português e está disponível online[1]

Na Tabela 1, tem-se alguns exemplos de pares de frases Tupi-Português.

Frase em Tupi	Frase em Português
Abá o-pytá 'y kuá-pe.	Uns índios ficam na enseada do rio.
Abá-pe o-só siri 'y-pe?	Quem vai para o rio dos siris?
Aîpó peró o-nhemokyrirî. Abá nhe'enga o-îkoé-eté.	Aquele português cala-se. A língua dos índios é muito diferente

Tabela 1: Exemplo de pares de frases Tupi-Portugûes

3.2 Modelagem dos Dados

Para se trabalhar com os dados e fazer a construção do algoritmo, foi utilizado a plataforma *Google Colab*, pois esta plataforma disponibiliza processamento partir de GPUs de forma gratuita, possibilitando um alto poder de processamento. Porem para se implementar e realizar os paços descritos neste artigo, poderá se utilizar qualquer IDE da linguagem python de sua preferência

Além da coleta dos pares de frases, para ser possível utilizar o *dataset* em conjunto com o algoritmo de tradução, algumas etapas foram necessárias para fazer a modelagem dos dados, tais etapas são descritas a seguir.

3.2.1 Importação das Bibliotecas e Carregamento do Dataset

Para se trabalhara com os dados é preciso primeiramente possuir as ferramentas necessárias, para isto fazemos a importação de algumas bibliotecas da linguem Python que nos permite fazer a importação dos dados, assim como o tratamento dos mesmos.

Para realizar a importação do *dataset*, foi extraído as frases de cada língua para dois arquivos de texto diferentes contendo as frases coletadas, ao final de cada frase existe uma quebra de linha para que cada frase esteja em uma linha distinta do arquivo, desta forma podemos referenciar cada frase com o seu respectivo par da outra língua. Após este processo foi feita à importação dos arquivos de texto para a plataforma de desenvolvimento.

3.2.2 Limpeza dos Dados

Nesta etapa foram realizadas a implementação de algumas expressões regulares para fazer a remoções de algumas inconsistências existentes na base de dados, como a remoção de alguns caracteres especiais por exemplo. Nesta etapa também foi definido que a cada quebra de linha, será considero o fim e início de uma nova frase.

[1] **https://docs.google.com/spreadsheets/d/1Tueqm4-_PwNt4aHTbNETtWgGJIMXBJ-SwdscAggJT9M/edit?usp=sharing**

3.2.3 Tokenização

Durante a etapa de *Tokenização* é realizado a transformação de cada palavra da base de dados de texto para número, a partir deste momento cada palavra será identificada por um número distinto. Este processo se faz necessário pois os computadores não conseguem entender diretamente os caracteres em formato de texto, apenas entendem números.

3.2.4 Padding e Batches

Na etapa de *Padding* e *Batches* foi construído um filtro para aceita frases apenas com um tamanho máximo definido, para o dataset Tupi-Portugês definimos um tamanho máximo de 50 caracteres, ao definir o máximo de caracteres aceitos, também será necessário alterar o tamanho das frases que possuem um tamanho menor do que o máximo estipulado. Para frases com tamanho menor do que 50 caracteres, e realizado o preenchimento dos espaços vazios com zeros para deixar as frases com tamanhos iguais.

3.3 Implementação do Algoritmo Transformer

O após o *dataset* ter atingido um tamanho razoável de pares de frases, foram realizados implementações e testes de algoritmos de tradução automática, para se saber qual algoritmo de tradução se encaixaria melhor e, teria melhores resultados com o *dataset*. O algoritmo escolhido foi o algoritmo da arquitetura *Transformer*.

A escolha do algoritmo foi baseada na quantidade de matérias e documentação disponíveis que detalham a arquitetura, desta forma vindo a facilitar o entendimento e a implementação do algoritmo. Para se implementar o algoritmo foi utilizado a linguagem de programação Python, já que as bibliotecas da linguagem facilitam a implementação dos algoritmos de aprendizagem de máquina. Para se implementar foi seguido alguns passos que serão descritos nos sub tópicos a seguir.

3.3.1 Embedding

Nesta etapa criamos um mecanismo de *Positional Encoding*, que é responsável por identificar em qual posição a palavra se encontra dentro de uma determinada frase que chega como entrada para o algoritmo, isto se faz necessário para que seja possível identificar em qual posição uma determinada palavra se encontra dentro de uma frase, assim como identificar quais as palavras anteriores e posteriores.

3.3.2 Mecanismo de Atenção

O mecanismo de atenção ou camada de atenção, é a etapa pela qual se é atribuída para cada palavra um grau de importância dentro do contexto no qual ela está inserida, na arquitetura *Transformer* existem dois mecanismos de atenção, o *Scaled Dot-Product Attention* e o *Mult-Head Attention*.

Durante o *Scaled Dot-Product Attention*, se é atribuído um grau de importância para a palavra baseado na própria frase em que ela está inserida. Neste processo se é realizado duas

copias da mesma frase que foi inserida como entrada, logo após são realizados cálculos de similaridade que relacionam uma determinada palavra com as outras palavras da mesma frase. Ao final é gerada uma nova frase com pesos de importância atribuído para cada palavra.

Já o mecanismo de atenção *Mult-Head Attention,* trabalha com a atenção da frase que é passada como entrada juntamente com a sua respectiva frase no outro idioma que é passada como possível saída, com isto e criado uma ligação entre as duas frases em seus respectivos idiomas.

3.3.3 Encoder e Decoder

Na arquitetura *Transformer* assim como na *Seq2Seq,* existem um mecanismo de *Encoder* e outro de *Decoder*. O *encoder* e o *decorder* são formados principalmente pelos mecanismos de atenção descritos no tópico anterior.

O *Encoder* possui apenas o mecanismo de *Scaled Dot-Product Attenti*on que faz processamentos em cima das frases de entradas. Já o *Decoder* além de possuir o mecanismo *de Scaled Dot-Product Attention,* também possui o mecanismo de *Mult-Head Attention,* que é responsável por relacionar as frases inseridas como entrada e as frases passadas como possíveis saídas.

3.3.4 Transformer

Nesta etapa é realizada a ligação entre todos os mecanismos criados anteriormente, para isso criamos algumas camadas, variáveis e funções utilizando a linguagem Python, para que seja possível com que possamos passar os dados de entradas e as projeções entre os mecanismos e funções de processamento construídos para a arquitetura.

3.4 Treinamento do Algoritmo

Nesta etapa foi construído uma parte do algoritmo que é responsável por estanciar os elementos da arquitetura, passando para ela as frases de entradas e as possíveis frases de saída, já que este é um problema de aprendizagem supervisionada, onde sabemos qual é entrada e qual seria a saída desejada. Durante o treinamento as frases do idioma que desejamos traduzir são passadas para a parte do *Encoder,* e as frases do idioma alvo são passadas para o *Decoder*.

Um dos principais parâmetros criados para se fazer o treinamento, foi o parâmetro de *dropout_rate,* este parâmetro é necessário para que durante o treinamento aconteça a retirada de alguns pares de frases da base de dados, para serem utilizados exclusivamente para os testes, desta forma evitamos que o algoritmo aprenda apenas a traduzir as frases existentes no *dataset*. Para o treinamento foi crido um *loop* de épocas, em que se é definido quantas vezes o treinamento se repetirá utilizando os dados do *dataset*.

3.4.1 Métricas de Medição

Foram criados alguns paramentos que são de uso comum em etapas de treinamento em algoritmos de aprendizagem máquina, para assim podemos acompanha o desempenho do

algoritmo e tenhamos um resultado ao final do treinamento, sem precisarmos fazer testes manualmente.

Foi criado uma função de erro (*loss*), que simplesmente compara as respostas corretas que já existem na base de dados, com as previsões fornecidas pelo algoritmo, no final do cálculo do erro e calculado um media entre os acertos e erros gerados pelo algoritmo.

Foram criadas duas variáveis para armazer os valores da média de erro e da média de assertividade durante o treinamento. A variável "*Loss*" armazenara a média de erros, já a variável "*Accuracy*" armazena a média de assertividade do algoritmo.

4. RESULTADOS

Nessa seção, serão apresentados os resultados obtidos a partir da implementação do algoritmo *Transforme* utilizando duas fontes de dados, a primeira sendo um *dataset [1]* de exemplo, do idioma inglês para português de Portugal, tal *dataset* será utilizado para comprovar o funcionamento do algoritmo de tradução. O tamanho total deste *dataset* é de 1960408 pares de frases, para ser possível a realização do teste com essa base, foi realizada uma limitação para se usar apenas 208990 pares de frases.

4.1 Resultados do Dataset Inglês-Português

Após o treinamento com este *dataset* chegamos a um *Accuracy* de 50% de acerto, Este é um resultado considerado aceitável, já que foi utilizando uma pequena fração do *dataset*. Abaixo temos alguns resultados de traduções pós treinamento do algoritmo.

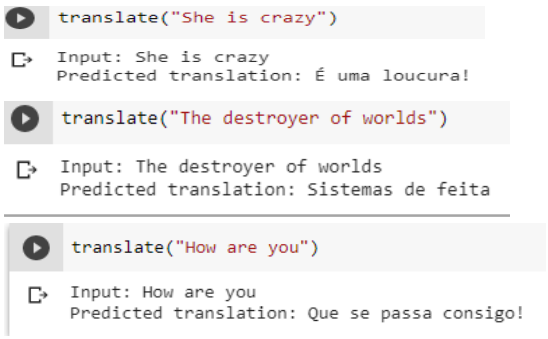


Figura 2: exemplos de tradução do algoritmo transformer com a base inglês-portugûes

Na figura 2 podemos ver alguns exemplos de traduções que foram geradas pelo algoritmo *Trasformer*. Como podemos ver, a maioria das traduções não são tradução 100% corretas, mas na maioria das vezes as traduções têm algum sentido além de estarem no idioma alvo, em alguns casos as traduções são aceitáveis, como no último exemplo.

4.2 Resultados do Dataset Tupi-Português

[1] **https://www.statmt.org/europarl/**

Diferentemente do *dataset* Inglês-Português, no *dataset* Tupi-Português temos atualmente pouquíssimos pares de frases, no momento da escrita deste artigo contamos com 2 mil pares de frases. Porém, apesar da quantidade limitada de pares de frases, conseguimos atingir uma *Accuracy* de 12% de assertividade. Tal porcentagem pode parecer insignificante a uma primeira vista, mas como não estamos tratando de um problema binário, este é um valor aceitável para um *dataset* de apenas 2 mil pares de frases, principalmente se fizermos uma relação com o *dataset* Inglês-Português.

Isto acontece por uma característica do idioma Tupi, pois neste idioma temos relativamente menos palavras do que no idioma Português, fazendo com que as palavras e frases em Tupi se repetiam mais frequentemente durante o processo de treinamento, reforçando a aprendizagem do algoritmo. A seguir temos alguns exemplos de traduções.

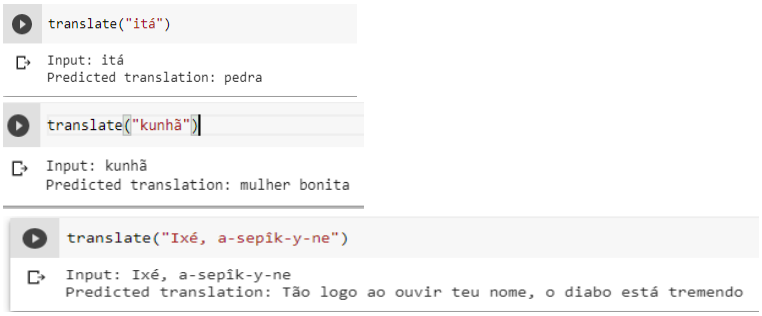


Figura 3: exemplos de tradução do algoritmo transformer com a base inglês-portugûes

Na Figura 3, é apresentado alguns exemplos de traduções, o algoritmo treinado consegue acertar algumas palavras, no primeiro exemplo passamos a palavra itá, que significa pedra em Tupi, e algoritmo conseguiu traduzir corretamente. No segundo exemplo passamos a palavra kunhã, que significa mulher, o algoritmo traduziu para “mulher bonita”.

Como o algoritmo foi treinado utilizando pares de frases, normalmente ele irá tentar encaixar as traduções dentro de um contexto, por isto muitas vezes ele irá receber uma pequena entrada e devolverá uma saída com bem mais palavras, como no último exemplo, que a tradução correta seria apenas “Eu o verei”.

5 CONCLUSÃO

Como foi visto, o algoritmo de tradução funciona relativamente bem e cumpriu o seu objetivo juntamente com o *dataset* desenvolvido neste trabalho, porém fica evidente que o mesmo necessita de uma quantidade razoável de pares de frases para gerar boas traduções. No entanto podemos comprovar que apenas com 2 mil pares de frases, já foi capaz de gerar algumas traduções aceitáveis, mostrado todo o potencial da arquitetura *Transformer* para traduções.

O *dataset* Tupi-Português ainda se encontra com poucos pares de frases, mas continua a crescer a cada dia, obtendo cada vez melhores resultados nas traduções do tupi para o português, indicando que com um pouco mais de pares de frases já seria possível ter boas traduções.

REFERÊNCIAS

1. Campos, Vivian, André Luiz da Silva, and Douglas Rodrigues da Silva. "A formação do povo brasileiro e o reconhecimento efetivo da diversidade: Cultura, educação e ações afirmativas em prol de uma sociedade reflexiva." *Revista Magistro* 1.15 (2017).
2. Ferreira, Isabel Alves. "A Questão indígena na sala de aula: desafios e contradições do ensino e um estudo de caso na Escola de Educação Básica Helena de Oliveira-Pilar/AL." (2019).
3. Ana Paula de Araujo. História da língua portuguesa no Brasil 2019. <https://www.infoescola.com/portugues/historia-da-lingua-portuguesa-no-brasil>, Acesso em: 04, julho, 2020
4. Rodrigues, A. D., & Silva, J. G. de G. da. Línguas Tupí em Rondônia e na Bolívia Oriental. *Revista Brasileira De Linguística. 2018*
5. MONTEIRO, John Manuel. Tupis, tapuias e historiadores: estudos de historia indigena e do indigenismo. 2001. 233f. Tese (livre-docência) - Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas, Campinas, SP.
6. Caseli, Helena de Medeiros. "Tradução Automática: estratégias e limitações." *Domínios de Linguagem* 11.5 (2017)
7. PEREIRA, Luís Moniz. Inteligência Articial Mito e Ciência. São Paulo, 2005. Disponível em: http://centria.fct.unl.pt/lmp/publications/online-papers/ia-mito.pdf> Acesso em: 25, julho, 2020.
8. McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5,** 115–133 (1943). https://doi.org/10.1007/BF02478259.
9. Vieira, Flávio Henrique Teles, Rodrigo Pinto Lemos, and Luan Ling Lee. "Alocação Dinâmica de Taxa de Transmissão em Redes de Pacotes Utilizando Redes Neurais Recorrentes Treinadas com Algoritmos em Tempo Real." *IEEE Latin America* 1.1 (2003).
10. Moura, Gustavo Bennemann de. *Redes neurais recorrentes para a classificaçao de estruturas retóricas*. MS thesis. Universidade Estadual de Maringá., 2018.
11. Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).
12. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.
13. KOEHN, Philipp. European parliament proceedings parallel corpus 1996-2011. 2011. < https://www.statmt.org/europarl>Acesso em: 07, julho, 2020
14. Navarro, Eduardo de Almeida. "Método moderno de tupi antigo: a língua do Brasil dos primeiros séculos." *São Paulo: Editora Vozes* (1998).