UNIVERSIDADE FEDERAL DO ACRE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

UMA ANÁLISE DO IMPACTO DO TAMANHO DO CONTEXTO EM SISTEMAS CONVERSACIONAIS BASEADOS EM *DEEP LEARNING*

RIO BRANCO

2019

AMÉLIA ALICE CARDOSO FEITOSA

UMA ANÁLISE DO IMPACTO DO TAMANHO DO CONTEXTO EM SISTEMAS CONVERSACIONAIS BASEADOS EM *DEEP LEARNING*

Monografia apresentada como exigência parcial para obtenção do grau de bacharel em Sistemas de Informação da Universidade Federal do Acre.

Prof. Orientador: Dr. Raoni Simões Ferreira

RIO BRANCO

2019

TERMO DE APROVAÇÃO

AMÉLIA ALICE CARDOSO FEITOSA

UMA ANÁLISE DO IMPACTO DO TAMANHO DO CONTEXTO EM SISTEMAS CONVERSACIONAIS BASEADOS EM *DEEP LEARNING*

Esta monografia foi apresentada como trabalho de conclusão de Curso de Bacharelado em Sistemas de Informação da Universidade Federal do Acre, sendo aprovado pela banca constituída pelo professor orientador e membros abaixo mencionados.

Compuseram a banca:

Prof. Dr. Raoni Simões Ferreira
Curso de Bacharelado em Sistemas de Informação

Prof. Dr. Manoel Limeira de Lima Júnior Almeida
Curso de Bacharelado em Sistemas de Informação

Prof. Dr. Luiz Augusto Matos da Silva
Curso de Bacharelado em Sistemas de Informação

Rio Branco - AC, 15 de fevereiro de 2019.

Dedico esta monografia a minha família, amigos e professores.

AGRADECIMENTOS

Primordialmente agradeço a Deus por ter me dado sabedoria, inteligência e saúde durante esta caminhada.

Agradeço a minha *Mother*, por sempre me dar forças e mostrar que posso fazer ou ser o que eu quero. Especialmente pelos conselhos, ensinamentos, amor, amizade e momentos dedicados. Ao me guiar por todos esses anos, eu não sei se ela chegou a perceber que a pessoa que eu mais queria ser era ela. Obrigada *Mother*, minha admiração.

Ao meu pai e meu irmão, meus sinceros agradecimentos pelo apoio e carinho, além de não medirem esforços para que eu chegasse até esta etapa da minha vida.

A todo o restante da minha família, mas principalmente minhas tias que pagaram meus estudos que foram de suma importância para que eu chegasse a graduação com educação de qualidade.

A pessoa com quem eu compartilhei momentos incríveis e que tive a honra de conhecer durante a graduação, obrigada pelo carinho, paciência e por ter tido capacidade de me trazer paz em momentos de ansiedade.

Agradeço a todos os amigos que fiz durante a graduação, especialmente aqueles que levarei pelo resto da vida, Antônia Gabriela, Mariana Xavier, Michele

Ascoli e Leôncio Carioca, pois eles alegraram meus dias, proporcionaram momentos inesquecíveis e aliviavam a pressão que surgia às vezes. No entanto, não posso deixar de citar o trio parada dura nas maratonas de programação, Time ERROR 404 desde 2015 para sempre.

Agradeço grandemente pela orientação do professor Raoni Simões Ferreira que teve paciência, dedicação, mas sobretudo, por ter aceitado mergulhar comigo na construção deste trabalho, pois eu não imaginava que chegaríamos a este tema, foi muito mais além do que pensei, foi melhor, muito obrigada professor.

Por fim, mas não menos importante, agradeço a instituição pelo ambiente e oportunidades que consegui usufruir ao longo do curso, como as monitorias. Agradeço a todos os professores que compartilharam seus conhecimentos durante a graduação e que contribuíram para a minha formação profissional. Em particular, agradeço ao professor Manoel Limeira de Lima Júnior Almeida, pela confiança e ensinamentos. Mas também, a alma (no sentido bom) da coordenação, Socorro Pontes que sempre me ajudou sem medir esforços quando precisei.

A todos que direta ou indiretamente fizeram parte da minha formação, muito obrigada.

"Um computador mereceria ser chamado de inteligente se pudesse enganar um humano acreditando que era humano."

Alan Turing

RESUMO

Os sistemas conversacionais estão integrados em diversos serviços que utilizamos no dia a dia, no entanto, ainda existe um grande desafio em aberto na construção de tais sistemas, que é desenvolver métodos que aprendam automaticamente, a partir de um repositório de diálogos, para que forneçam a melhor resposta para o usuário humano. Trabalhos existentes utilizam a abordagem de recuperação das melhores respostas dado um contexto de conversação com múltiplos turnos de interação, para construir sistemas conversacionais a partir dos textos dos diálogos. O contexto da conversa tem impacto direto na qualidade das respostas fornecidas. Nesse cenário, o presente trabalho teve como objetivo construir o modelo de *Deep Learning* Dual Encoder LSTM e o modelo de Espaço Vetorial que são comumente usados na literatura científica para desenvolvimento de sistemas conversacionais, e avaliar o impacto do tamanho do contexto na qualidade das respostas fornecidas por estes modelos. Os resultados mostraram que ao selecionar a melhor resposta das 10 respostas candidatas (i.e., R@1) o modelo LSTM teve pior desempenho em R@1 a medida que o tamanho do contexto usado para avaliá-lo crescia, enquanto que o modelo de Espaço Vetorial apresentou desempenho superior em R@1 quando submetido as mesmas condições do LSTM. Por outro lado, quando ambos foram avaliados sobre um conjunto de diálogos composto por contextos de tamanhos aleatórios, LSTM conseguiu superar o desempenho do Espaço Vetorial.

Palavras-chave: Sistemas Conversacionais, *Deep Learning*, Inteligência Artificial, Redes Neurais, Aprendizado de Máquina, contexto.

ABSTRACT

Conversational systems are integrated into a variety of services we use on a day-to-day basis; however, there is still a major open challenge in building such systems, which is to develop methods for automatically learning from dialogue repository to provide the best response for a human user. Existing works use the responses retrieve approach based on a conversation context of multiple turns to build conversational systems from text dialogues. The conversation context has a direct impact on the quality of responses provided by these methods. In this scenario, the objective of this work was to construct the Deep Learning Dual Encoder LSTM model and the Vector Space model, both are commonly used in the scientific literature for the development of conversational systems, as well as evaluate and analyze the impact of the context size over the quality of responses provided by these models. The results showed that by considering the first from 10 responses (i.e., R@1), the LSTM model presents the performance worse than the Vector Space Model as the context size increased. On the order hand, when both were evaluated on a dataset composed of random context sizes, LSTM was able to overcome the Vector Space performance.

Key-words: Conversational Systems, Deep Learning, Artificial Intelligence, Neural Networks, Machine Learning, context.

LISTAS DE FIGURAS

LISTAS DE QUADROS

LISTAS DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (IA), por definição, é a inteligência exibida pelas máquinas, seu objeto é estudar os agentes inteligentes que recebem percepções do ambiente e executam ações (NORVIG; RUSSELL, 2013). Um agente inteligente pode ser definido, por exemplo, como um computador que é capaz de simular realisticamente a comunicação humana.

Desse modo, a IA Conversacional é uma área de pesquisa ativa e de interesse crescente, que proporciona uma interface de interação fácil entre humanos e computadores. Devido aos seus potenciais promissores e atraentes valores comerciais, os sistemas conversacionais atuam como suporte técnico em lojas *online*, bancos, companhias aéreas, operadoras de telefonia e cartões de crédito, e assistentes de voz quando integrados as funções dos *smartphones*, para citar apenas alguns exemplos. Por esses motivos, é uma área que explicitamente pede contribuições de estudos de conversação (YAN, 2018).

O desenvolvimento e a aplicação desses sistemas em diversos domínios têm atraído a atenção da comunidade acadêmica que tem enxergado como uma oportunidade de melhor entender os mecanismos de diálogos em linguagem natural entre humanos e computadores (WANG et al., 2013).

Nesse contexto, pesquisas sobre modelagem de conversações, que começaram há mais de meio século, foram renovadas principalmente com: (i) o

surgimento de bases de dados públicas de diálogos; (ii) aumento substancial do poder computacional para experimentação desses sistemas e (iii) de novas abordagens de desenvolvimento de tais sistemas, como aquelas baseadas em técnicas das áreas de recuperação de informação e de aprendizado de máquina (LOWE et al., 2015; SAKAI et al., 2015).

Nesta monografia, foi considera do o estudo do problema de conversação e a avaliação de dois modelos usados para implementar sistemas conversacionais: o primeiro que modela os diálogos usando uma técnica clássica da área de recuperação de informação e o segundo que modela os diálogos considerando um modelo de aprendizado de máquina supervisionado baseado em *Deep Learning*.

1.1 JUSTIFICATIVA DA PESQUISA

A tarefa de conversação em linguagem natural entre humano e computador é considerada um dos problemas mais desafiadores da Inteligência Artificial. Para que essa conversa aconteça, é desejável que tal computador seja capaz de fornecer respostas similares à que um humano forneceria em um diálogo. Isso é possível, se tal computador é dotado de um sistema inteligente que implementa mecanismos de raciocínio e de aprendizado com base no conhecimento de um domínio (como, por exemplo, um sistema conversacional usado para reservar passagens aéreas), além de recursos que apoiam a compreensão da linguagem humana (WANG et al., 2013; WU et al., 2016).

O Teste de Turing, por exemplo, adota justamente essa tarefa como referência para verificar a capacidade de um computador exibir comportamentos inteligentes, equivalentes a um ser humano durante uma sessão de diálogo, composta por diversas rodadas de interações, incluindo afirmações feitas pelo agente humano e as respostas produzidas pelo agente computador (SAYGIN; CICEKLI; AKMAN, 2000).

Neste trabalho, foi considerado o problema de conversação em um cenário onde o repositório de diálogos é não estruturado (semelhante aquele encontrado em

mídias sociais, *e-mails* ou fóruns de discussões, onde não há uma lógica *a priori* do fluxo de troca de informações), individualizado (isto é, entre um agente humano e um agente computador) com diversas rodadas de interação ou multi-turno (LOWE et al., 2015).

Assim, ao lidar com este problema, as soluções existentes adotam a abordagem chamada de recuperação. A ideia é manter um grande repositório de diálogos e, a partir deste repositório, são desenvolvidos sistemas de conversação para recuperar respostas com base em um contexto do diálogo que representa todas as frases ditas até um determinado ponto da conversa.

Para ilustrar o problema proposto, considere um contexto *C* referente a uma conversa, o sistema de conversação busca no repositório e retorna a resposta mais adequada dentro de um conjunto de respostas candidatas. Note que as respostas que estão no repositório foram originalmente dadas como respostas para algum outro contexto que não seja *C*, mas que podem ser reutilizadas como resposta aceitável para o contexto *C*.

Uma direção de pesquisa que tem sido investida na tarefa de conversações multi-turnos, é a que implementa esses sistemas usando *Deep Learning*, onde um sistema de conversação inteligente (i.e, o agente computador) é treinado com parte dos dados do repositório de diálogos, com o objetivo de aprender a classificar as respostas e retornar aquela cuja a probabilidade de ser classificada como tal, dentre as possíveis candidatas, seja aquela que mais se aproxima a de um ser humano, quando este fosse requisitado a fornecer uma resposta durante uma sessão de diálogo.

Para resumir, *Deep Learning*, é uma abordagem para lA. Especificamente, é um tipo de aprendizado de máquina, que alcança grande poder e flexibilidade ao permitir que o computador aprenda conceitos complicados, construindo-os a partir de conceitos mais simples (GOODFELLOW; BENGIO; COURVILLE, 2016). Tais conceitos são obtidos a partir dos exemplos de diálogos do repositório e são amplificados à medida que navegam pelas camadas de uma rede de *Deep Learning* (LECUN; BENGIO; HINTON, 2015).

Sistemas conversacionais baseados em *Deep Learning*, fazem uso de modelos neurais sofisticados, portanto, vários modelos têm sido empregados recentemente neste contexto (KADLEC; SCHMID; KLEINDIENST, 2015; LOWE et al., 2015, 2017). Trabalhos relacionados a esta monografia e que usam tais modelos são sumarizados na Seção 1.2.

Dentre os estudos realizados nesta direção de pesquisa, é importante ressaltar que os trabalhos recentes reportam implementação de sistemas treinados com diálogos com contextos de tamanhos aleatórios, isto é, diálogos com múltiplos turnos de interação. Um aspecto não reportado por esses trabalhos e seguido nesta monografia, foi de avaliar o impacto na qualidade das respostas, quando um sistema de conversação é treinado com diálogos de tamanhos fixos de turnos.

Em outras palavras, entender se o tamanho do contexto no treinamento (por exemplo, um sistema treinado com diálogos com contexto de um, dois ou mais turnos de interação), causa algum impacto na recuperação de uma resposta. Portanto, neste trabalho, procura-se construir e avaliar dois algoritmos usados em sistemas de conversação, para selecionar respostas em diálogos multi-turno.

Para tanto, será usada uma implementação baseada em um modelo de *Deep Learning* apresentada no trabalho de Lowe et al. (2015), chamada Dual Encoder LSTM*,* e uma segunda implementação baseada em um algoritmo heurístico clássico da área de Recuperação de Informação, chamada de modelo de Espaço Vetorial, como *baseline*.

1.2 TRABALHOS RELACIONADOS

Nesta seção, é feito um breve resumo dos principais trabalhos relacionados a esta monografia, que estudam a tarefa de recuperação de respostas para conversas multi-turnos. Porém, não foram encontrados trabalhos que analisassem o impacto do tamanho do contexto sobre a qualidade das respostas.

Lowe et al. (2015) levantam uma hipótese que devido à falta de conjuntos de dados suficientemente grandes para conversas multi-turnos, impediu o progresso no desenvolvimento de sistemas de conversação. Dessa forma, o artigo propõe um *corpus*, denominado Ubuntu Dialogue Corpus, que consiste em registros de interações em salas de bate-papo da rede *Freenode Internet Relay Chat (IRC)* relacionadas ao sistema operacional Ubuntu. O artigo também apresenta arquiteturas de aprendizado adequadas para analisar o *corpus*, variando da abordagem *Term Frequency–Inverse Document Frequency* (TF-IDF) (i.e., baseada no modelo de Espaço Vetorial) a arquiteturas neurais mais sofisticadas, incluindo *Recurrent Neural Networks* (RNN) e *Long Short-Term Memory* (LSTM), reportando o desempenho de cada uma. A arquitetura LSTM superou a RNN e a TF-IDF em todas as métricas de avaliação.

Em uma extensão do trabalho referenciado anteriormente, Lowe et al. (2017) analisaram sistemas de conversação baseados em redes neurais, treinados de uma forma completa, usando uma versão atualizada do Ubuntu Dialogue Corpus (2.0). O trabalho inclui resultados sobre modelos baseados em geração e uma avaliação mais ampla dos modelos baseados em recuperação, bem como, uma análise qualitativa das respostas dos modelos generativos e erros de classificação feitos pelo modelo Dual Encoder LSTM. Também discutiram que, embora os modelos generativos e de recuperação sejam promissores, eles ainda cometem muitos erros óbvios e há espaço significativo para melhorias.

Kadlec, Schmid e Kleindienst (2015) implementam três arquiteturas: *Convolutional Neural Networks* (CNN), *Long Short-Term Memory* e *Bi-Directional Long Short-Term Memory* (Bi-LSTM). Essas arquiteturas são avaliadas separadamente e em conjunto, utilizando o Ubuntu Dialogue Corpus. O melhor desempenho foi obtido com o conjunto das arquiteturas. Os autores também experimentam com diferentes quantidades de dados de treinamento, evidenciando que a arquitetura CNN supera a LSTM e a Bi-LSTM se o conjunto de dados de treinamento é pequeno, enquanto que a LSTM e Bi-LSTM funcionam melhor com dados suficientes.

Todavia, nesta monografia, a arquitetura de rede neural adotada foi a *Long Short-Term Memory* e como *baseline* foi selecionada a TF-IDF baseada no modelo de

Espaço Vetorial, pois existem implementações disponíveis publicamente, enquanto as outras arquiteturas não possuem.

1.2 OBJETIVOS DA PESQUISA

Nesta seção serão descritos os objetivos geral e específicos da pesquisa.

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo construir o modelo de *Deep Learning* Dual Encoder LSTM e o modelo de Espaço Vetorial, que são utilizados para desenvolvimento de sistemas conversacionais, e avaliar o impacto do tamanho do contexto na qualidade das respostas fornecidas por estes modelos.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral da pesquisa, foi necessário alcançar os seguintes objetivos específicos:

a) Estudar o problema de conversação multi-turnos e as soluções mais recentes para resolução deste problema.

b) Estudar os princípios de um modelo de *Deep Learning* chamado Dual Encoder LTSM e sua aplicação no problema de conversação multi-turnos.

c) Estudar os princípios do modelo de Recuperação de Informação chamado Modelo de Espaço Vetorial e sua aplicação no problema de conversação multi-turnos.

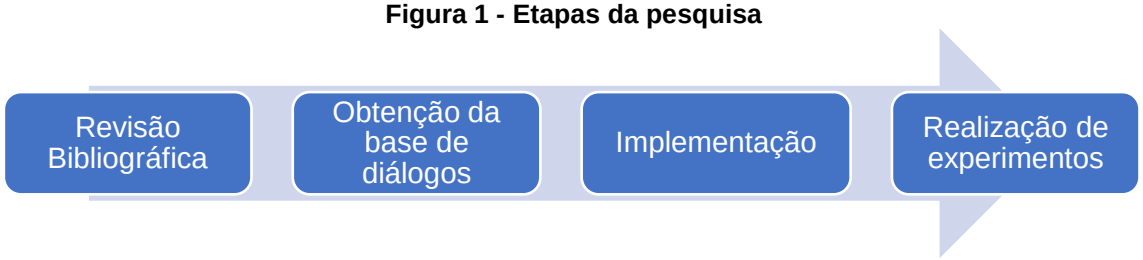
d) Construir os modelos a partir da implementação dos algoritmos estudados, avaliar e analisar a qualidade das respostas sugeridas por eles, sobre um repositório de diálogos, usando métricas objetivas comumente utilizadas para medir o desempenho de sistemas de conversação.

1.3 METODOLOGIA

De acordo com Wazlawick (2014), a pesquisa, no contexto científico, pode ser classificada conforme algumas características, como seus objetivos e procedimentos técnicos. No que diz respeito aos objetivos, esta pesquisa pode ser classificada como exploratória, uma vez que se enquadra em um estudo de caso que tem como objetivo avaliar o impacto do tamanho do contexto, sobre as respostas fornecidas por sistemas de conversação multi-turnos. Quanto aos procedimentos técnicos utilizados na avaliação, esta pesquisa pode ser classificada como experimental, pois foram realizados experimentos com o tamanho do contexto.

Em relação às etapas de pesquisa, este trabalho foi divido em quatro etapas, como mostra a Figura 1:

Figura 1 - Etapas da pesquisa



Fonte: Elaboração própria.

a) Revisão bibliográfica para seleção dos modelos e algoritmos (bem como suas implementações) baseados em Aprendizado de Máquina, Recuperação de Informação e Processamento de Linguagem Natural, usados para desenvolver sistemas de diálogos multi-turnos de recuperação de respostas.

b) Obtenção de uma base de diálogos pública para geração dos conjuntos de treinamento, validação e testes.

c) Adaptação das implementações dos algoritmos selecionados.

d) Realização dos experimentos, explanação das avaliações e comparações dos resultados com os algoritmos selecionados.

1.4 ORGANIZAÇÃO DA MONOGRAFIA

O restante deste trabalho está organizado em 3 capítulos. No Capítulo 2 são apresentados conceitos que fundamentam a pesquisa realizada, definindo o que são sistemas conversacionais e as formas de modelá-los, bem como as técnicas e algoritmos adotados para produzir esses sistemas e a métrica usada para avalia-los. No Capítulo 3 é apresentado o desenvolvimento do estudo de caso da pesquisa, descrevendo as bibliotecas, base de diálogos, metodologia de experimentação e os resultados obtidos. Por fim, o Capítulo 4 apresenta as considerações finais e recomendações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento de sistemas conversacionais remonta ao princípio da Ciência da Computação, quando os primeiros pesquisadores começaram a realizar significativos projetos para que computadores conseguissem interagir com pessoas através de linguagem natural. Mas, com os recentes avanços na tecnologia de computadores e no processamento de fala e linguagem, esses sistemas estão começando a parecer possíveis (NISHIDA et al., 2014).

A ciência que estuda a capacidade de sistemas conversacionais desenvolverem conversas naturais e contínuas entre humanos e computadores, é denominado Inteligência Artificial Conversacional. Sua próxima fronteira é alcançar um diálogo sustentável, coerente e envolvente (RAM et al., 2018).

Dessa forma, várias técnicas e algoritmos de subáreas da IA estão sendo aplicados no desenvolvimento de sistemas conversacionais, cujos conceitos serão abordados neste capítulo. A Seção 2.1 define o que são sistemas conversacionais e uma abordagem para tratar as conversas desses sistemas. A Seção 2.2 discorre sobre a área de Processamento de Linguagem Natural e suas técnicas. A Seção 2.3 apresenta o Modelo de Espaço Vetorial, adotado como *baseline*. A Seção 2.4 define o que é Aprendizado de Máquina Supervisionado. A Seção 2.5 denominada *Deep Learning* em Redes Neurais Artificiais, descreve desde os aspectos básicos de uma rede neural artificial, até arquiteturas de redes neurais artificiais sofisticadas, como as redes de *Deep Learning Recurrent Neural Networks* (2.5.1) e *Long Short-Term*

Memory (2.5.1.1), as quais são utilizadas no Modelo Dual Encoder LSTM, abordado na Seção 2.5.2. Por fim, a Seção 2.6 apresenta a métrica de avaliação utilizada neste trabalho.

2.1 SISTEMA CONVERSACIONAL

Um Sistema Conversacional (SC) é uma IA que pode simular uma conversa com um usuário em linguagem natural através de métodos auditivos ou textuais (HIJJAWI et al., 2014). Quanto mais longas essas conversas, mais complexo é de automatizá-las. Existem conversas de texto curto, onde o objetivo é criar uma única resposta para uma única entrada, representando uma interação ou turno de conversação (SHANG et al., 2016). Por exemplo, o sistema recebe uma pergunta específica de um usuário e a responde com uma resposta apropriada, como mostra a Figura 2. Por outro lado, temos longas conversas compostas por vários turnos (multi-turnos) e o sistema precisa acompanhar o que foi dito, conforme a Figura 3.

Figura 2 - Exemplo de conversa de um turno

User 1: Hey, what's the weather forecast for today?

User 2: The weather forecast shows light rain with a maximum of 27º and a minimum of 19º.

Fonte: Elaboração própria.

Figura 3 - Exemplo de conversa com múltiplos turnos

User 1: Hey, what's the weather forecast for today?

User 2: The weather forecast shows light rain with a maximum of 27º and a minimum of 19º.

User 1: So, should I bring an umbrella?

User 2: Yes, you probably want an umbrella today.

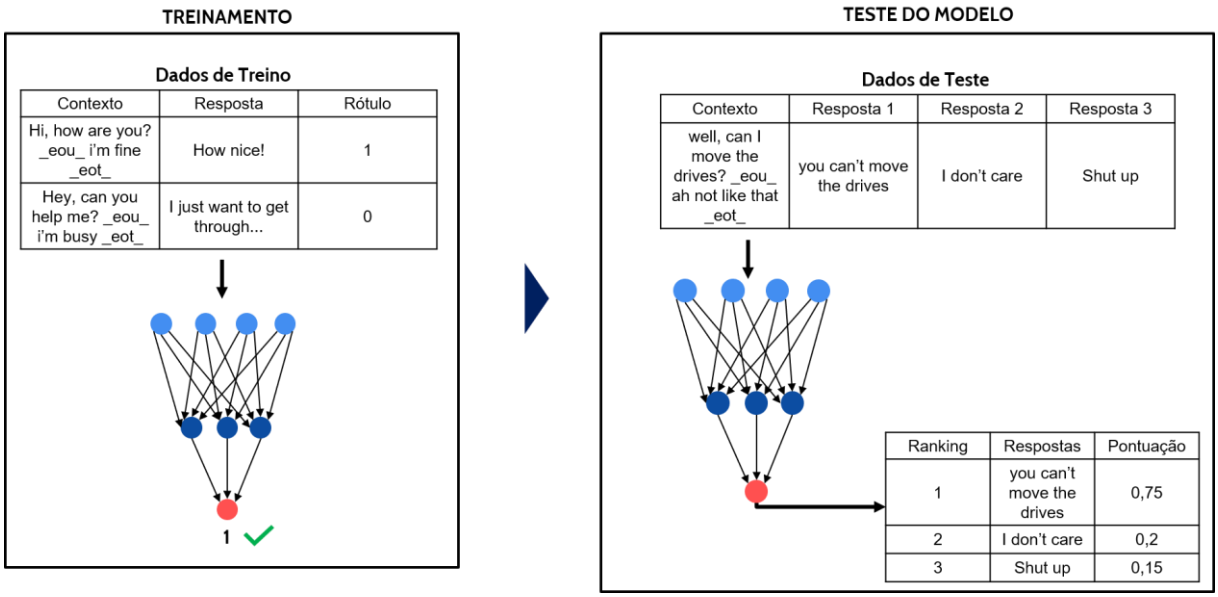
Fonte: Elaboração própria.

Assim, uma abordagem simples para tratar as conversas de um SC é utilizar a abordagem baseada em recuperação, que utiliza um repositório de diálogos com respostas pré-definidas e que por meio de algum tipo de heurística escolhe uma resposta apropriada com base na entrada (WANG et al., 2013). Sendo que, essa

heurística pode ser tão simples quanto uma correspondência de expressão baseada em regras ou tão complexa quanto um conjunto de algoritmos de Aprendizado de Máquina. Note que, essa abordagem não gera novos textos, mas apenas seleciona respostas a partir de um repositório de respostas fixo (JI; LU; LI, 2014).

A entrada de um algoritmo baseado em recuperação é um par contendo o contexto *C***,** que representa toda a conversa até um ponto do diálogo e a resposta esperada *R*. A saída do algoritmo é a melhor resposta de acordo com o contexto dado como entrada. Para tanto, o sistema fornece como saída uma lista ordenada em ordem decrescente das possíveis respostas com base em uma pontuação atribuída a cada uma e que representa as respostas candidatas mais apropriadas dado o contexto de entrada. Em geral, a resposta escolhida pelo sistema é aquela que está no topo desta lista (ou *ranking*). Sendo que, as várias respostas desta lista são válidas (porém, não necessariamente a esperada) e estão presentes no repositório de diálogos. Um sistema acerta na escolha se a primeira resposta deste *ranking* é aquela esperada.

Figura 4 - Arquitetura de um SC que usa aprendizado de máquina



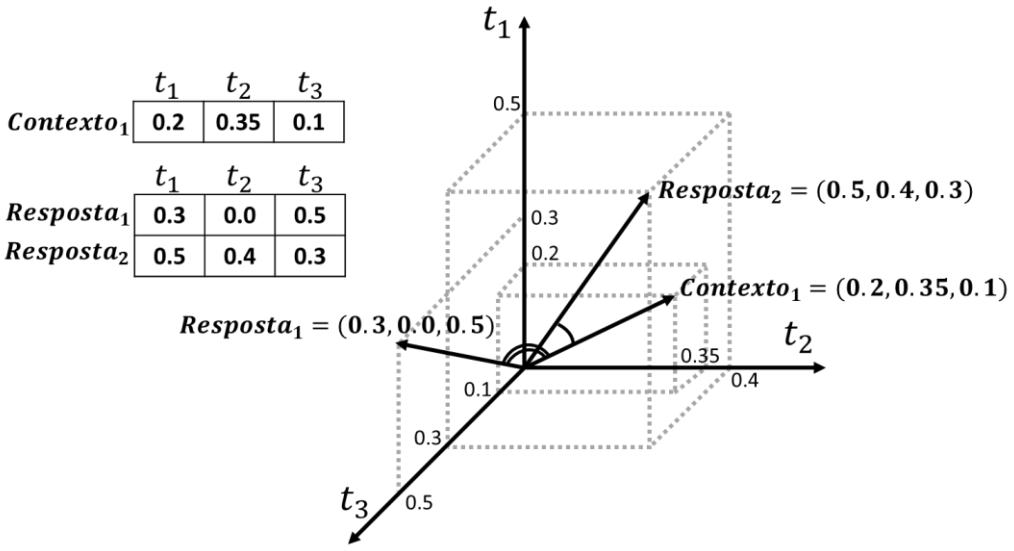
Fonte: Elaboração própria.

A Figura 4 ilustra a arquitetura de um SC baseado em recuperação que utiliza Aprendizado de Máquina. Essa arquitetura caracteriza-se por usar o repositório de exemplos de diálogos dividido em amostras de treino e teste. A amostra de treino contém pares (contexto, resposta) utilizados para treinamento de um algoritmo de aprendizado de máquina, que no caso deste trabalho, é um modelo baseado em *Deep*

Learning, enquanto que a de teste é usada para avaliar o modelo de aprendizado sobre pares não vistos antes pelo modelo.

Cada par da amostra de treino está rotulado com um valor binário: positivo se a resposta for a esperada (ou relevante ao contexto) ou negativa se a resposta não for a esperada (ou não relevante) ao contexto. Desta maneira, espera-se que o SC treinado seja capaz de generalizar a partir dos padrões encontrados na amostra de treinamento, o que possibilitará o sistema prever a pontuação das respostas e encontrar a mais apropriada para contextos da amostra de teste.

Figura 5 - Arquitetura de um SC que usa o modelo de espaço vetorial



Fonte: Adaptado de Ferneda (2012).

Por outro lado, o SC que usa modelo de similaridade entre documentos por meio do Modelo de Espaço Vetorial representado na Figura 5, busca medir a similaridade entre dois documentos, que aqui são representados pelo contexto e pelo conjunto de respostas candidatas. Como resultado, obtém-se um conjunto de respostas com suas respectivas pontuações que representam o grau de similaridade de cada resposta em relação ao contexto de entrada.

Ainda que não seja necessariamente verdade que uma boa resposta tenha muitas palavras comuns com o contexto, essa medida é frequentemente útil para encontrar respostas relevantes. Por exemplo, quando o contexto e a resposta contêm a palavra “chatbot”, é um indicativo de que eles tratam de assuntos semelhantes. Ao contrário do método que usa algoritmos de aprendizado de máquina, essa simples semelhança não requer aprendizado (WANG et al., 2013).

Portanto, a ideia básica do sistema conversacional baseado em recuperação é retornar uma pontuação (também chamada de *score*), a um conjunto de candidatas a respostas dado um contexto de entrada. Em diálogos com várias rodadas de interação ou multi-turnos, como é o caso deste trabalho, espera-se que a resposta com maior pontuação seja a resposta mais relevante para uma dada entrada fornecida.

Logo, faz-se necessário utilizar técnicas e algoritmos das áreas de aprendizado de máquina, processamento de linguagem natural e recuperação da informação para que um sistema possa produzir respostas relevantes. As demais seções deste capítulo descrevem como essas áreas podem ser empregadas e quais procedimentos são necessários para realizar recomendações em diálogos multi-turnos baseados em recuperação.

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

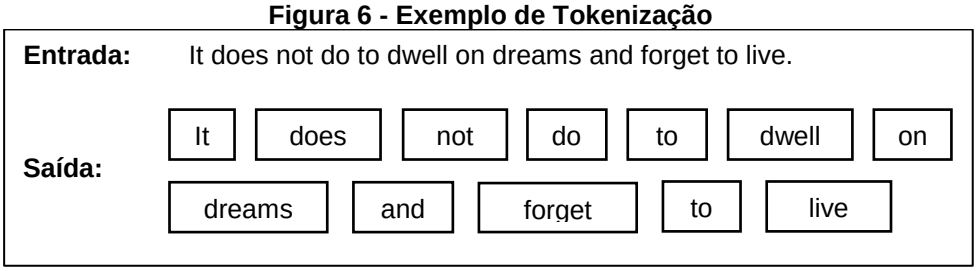
O Processamento de Linguagem Natural (PLN) emprega um conjunto de técnicas computacionais com o propósito de aprender, compreender e produzir conteúdo em linguagem humana (HIRSCHBERG; MANNING, 2015). Os computadores possuem limitações pois só podem processar o que está visível no texto, como por exemplo, a ocorrência de termos que pode ser interpretado como um indicador de proximidade semântica. Diferente dos humanos, computadores não conseguem realizar desambiguação e por vezes não são capazes de lembrar informações contextuais e, portanto, enfrentam problemas para interpretar (CAMBRIA; WHITE, 2014).

Por isso, para realizar o pré-processamento do conjunto de diálogos, foram utilizadas algumas técnicas de processamento de linguagem natural, dentre elas a *tokenização*, *stemming* e *lemmatization*. Estas técnicas são detalhadas nas subseções a seguir.

2.2.1 *Tokenization*

Segundo Korde (2012), a tokenização é o processo de cortar uma sequência de caracteres em pedaços, chamadas de *tokens* e ao mesmo tempo remover alguns caracteres, como pontuação. Esses *tokens* são uma instância de uma sequência de caracteres em algum documento específico que é agrupado como uma unidade semântica útil para processamento (MANNING; RAGHAVAN; SCHÜTZE, 2009).

Figura 6 - Exemplo de Tokenização



Fonte: Elaboração própria.

Um exemplo de tokenização de um documento é apresentado na Figura 6, onde um texto é fornecido como entrada e o processo de tokenização gera uma saída do mesmo documento com as palavras armazenadas em *tokens* e sem pontuação.

2.2.2 *Stemming* e *Lemmatization*

Por razões gramaticais, os textos utilizam formas diferentes de uma palavra, como *organize, organizes* e *organizing*. Portanto, o processo de *stemming* é basicamente a remoção do sufixo para gerar *stems* de palavras. Isso é feito para agrupar palavras que tenham o mesmo significado conceitual (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005), como é descrito nos exemplos do Quadro 1.

Quadro 1 - Processo de *stemming*

Palavra	Sufixo	Stem
Consigned	-ed	Consign
Consigning	-ing	Consign
Consignment	-ment	Consign

Fonte: Elaboração própria.

Lemmatization por outro lado, se refere a fazer as coisas corretamente, analisando se as palavras estão sendo usadas como verbos ou substantivos, utilizando um vocabulário e a análise morfológica da palavra, com o objetivo de remover as terminações flexionais, retornando assim as palavras à sua base ou a forma do dicionário, que é chamado de *lemma* (BALAKRISHNAN; ETHEL, 2014). Por exemplo, dadas as palavras *multiple*, *multiples*, *multiply* e *multiplied*, o *lemma* para cada uma delas é *multiple*.

Contudo, o objetivo de ambos, *stemming* e *lemmatization*, é "normalizar" as palavras para uma forma básica comum, o que é útil para muitas aplicações de processamento de texto, bem como um sistema conversacional. Assim, após pré-processamento, o conjunto de dados está pronto para ser submetido a algum modelo, bem como, um modelo clássico da área de Recuperação da Informação, denominado Modelo de Espaço Vetorial, discutido na Seção 2.3, ou um modelo de classificação de aprendizado de máquina supervisionado, abordado na Seção 2.4.

2.3 MODELO DE ESPAÇO VETORIAL

O Modelo de Espaço Vetorial é um modelo algébrico clássico da área de Recuperação da Informação, que representa documentos escritos em linguagem natural sob a forma de vetores de termos em um espaço multidimensional. Sua modelagem é baseada na suposição de que o significado de um documento pode ser entendido a partir dos termos que o constituem (BAEZA-YATES; RIBEIRO-NETO, 2013).

Nesta pesquisa, os contextos e as respostas são vistos como vetores de termos num espaço vetorial multidimensional, onde a distância vetorial é usada como medida de similaridade. Além disso, aos termos dos contextos e respostas são atribuídos pesos que especificam o tamanho e a direção de seus vetores de representação (CARDOSO, 2004).

O cálculo da similaridade é baseado no ângulo formado entre os vetores que representam o contexto (d) e a resposta (c), através da seguinte fórmula (CARDOSO, 2004):

$$sim(d, c) = \frac{\sum_{i=1}^{t} w_{id} x w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2}\, x \sqrt{\sum_{i=1}^{t} w_{iq}^2}} \quad (2.1)$$

Os pesos quantificam a importância de cada termo para as respostas (w_{iq}) e para os contextos (w_{id}) no espaço vetorial. O resultado da utilização destes pesos é a ordenação das respostas pelo grau de similaridade (pontuação) em relação ao contexto. Para a obtenção dos pesos w_{iq} e w_{id}, utiliza-se uma técnica chamada *Term Frequency-Inverse Document Frequency* (TF-IDF) (BAEZA-YATES; RIBEIRO-NETO, 2013).

O termo *term-frequency* é a frequência que uma palavra ocorre em um determinado contexto, enquanto que o termo *inverse document frequency* adiciona uma penalidade na frequência com que a palavra aparece em todo o repositório e mede a raridade da palavra no repositório de diálogos (LOWE et al., 2017). Assim, os pesos de cada palavra são resultantes do produto do TF e IDF da palavra, de acordo com a seguinte fórmula:

$$tfidf(w, d, D) = f(w, d)\ x\ \log \frac{|D|}{|\{d\ \in D : w\ \in d\}|} \quad (2.2)$$

Na fórmula 2.2, $f(w, d)$ indica a frequência da palavra w no contexto d e o denominador representa a quantidade de diálogos em que a palavra w aparece (LOWE et al., 2017).

Para realizar a classificação, os vetores TF-IDF são calculados primeiro para o contexto e para cada uma das respostas. Dessa forma, dado um conjunto de vetores de respostas candidatas, aquele que obtiver maior similaridade com o vetor de contexto, é então selecionado como resposta (LOWE et al., 2017).

Portanto, no Modelo de Espaço Vetorial, se um contexto e uma resposta contiverem palavras semelhantes, é mais provável que sejam um par correto. Entretanto, uma resposta não precisa necessariamente ser semelhante ao contexto para estar correta, além de que, uma grande desvantagem da técnica TF-IDF é que

ela ignora a ordem das palavras que carregam relações sintáticas e semânticas entre as palavras em um documento, e que podem ser importantes na tarefa de conversações multi-turnos (LI; MAK, 2016).

Assim, um modelo de classificação de aprendizado de máquina supervisionado, pode preencher essa deficiência do TF-IDF, tendo em vista que modelos de *Deep Learning,* como o Dual Encoder LSTM (Seção 2.5.2), lidam com dados sequenciais, como é o caso dos textos de diálogos, onde a sucessão das palavras é levada em consideração.

2.4 APRENDIZADO DE MÁQUINA SUPERVISIONADO

O aprendizado de máquina é uma subárea da Inteligência Artificial (IA) que estuda como os sistemas são capazes de aprender e melhorar automaticamente (MICHALSKI; CARBONELL; MITCHELL, 2013). É uma área com foco no desenvolvimento de sistemas que podem acessar dados e usá-los para aprender por si mesmos.

Cada vez mais, tais sistemas fazem uso de uma classe de técnicas de aprendizado de máquina, chamada *Deep Learning.* Entretanto, a forma mais comum de aprendizado de máquina, é o supervisionado, independentemente de ser *Deep Learning* ou não (LECUN; BENGIO; HINTON, 2015).

Segundo Baeza-Yates e Ribeiro-Neto (2013), algoritmos que necessitam de dados de treinamento rotulados como entrada para prever eventos futuros, são denominados algoritmos supervisionados e, portanto, necessitam de assistência humana para previsão.

No caso dos diálogos multi-turnos baseados em recuperação, os dados de treinamento são compostos de pares (contexto, resposta) rotulados indicando os rótulos corretos para os documentos dados. Esses dados de treinamento são utilizados para aprender uma função de previsão (ou classificação), a qual será usada para fazer predições de contextos dados ainda não vistos. Note que essa abordagem

só funciona se a função aprendida, aplica o que foi adquirido no passado a novos dados prevendo os rótulos destes com alta precisão (BAEZA-YATES; RIBEIRO-NETO, 2013).

Dessa forma, durante o treinamento de um modelo de *Deep Learning*, ele recebe um exemplo e produz uma saída na forma de um vetor de pontuações, uma para cada rótulo. Espera-se que o rótulo com pontuação mais alta seja o correto para o exemplo apresentado. Mas para se obter tal pontuação, é necessário realizar um cálculo através de uma função de custo, que mede o erro (ou distância) das pontuações de saída com relação à saída desejada, pois o conjunto de treinamento fornece ao modelo qual deve ser a saída esperada. Baseado nisso, o modelo realiza ajustes em seus parâmetros internos, geralmente chamados de coeficientes ou pesos, para minimizar o erro (LECUN; BENGIO; HINTON, 2015).

No entanto, um modelo pode apresentar sub aprendizado, quando os resultados são bons para o conjunto de treino, mas péssimos durante a fase de testes, ou ainda pode causar alto custo computacional durante o procedimento de treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para evitar que tais problemas ocorram, o treinamento pode ser interrompido antecipadamente, quando o desempenho do modelo parou de melhorar sobre um conjunto de validação, que é constituído de exemplos diferentes do conjunto de treino (PRECHELT, 2012).

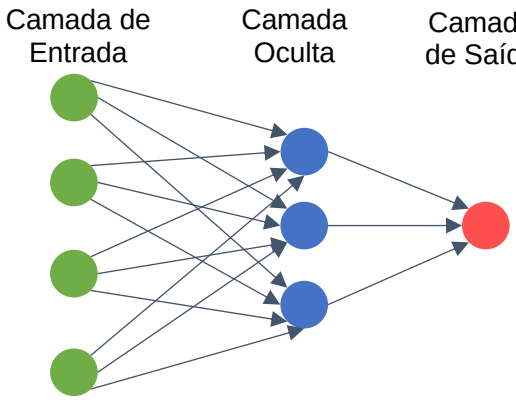
Após o treinamento, o desempenho do modelo é avaliado por meio de um novo conjunto de exemplos rotulados, denominado conjunto de testes. Essa avaliação contabiliza a porcentagem de exemplos classificados corretamente e a porcentagem de exemplos classificados incorretamente (LECUN; BENGIO; HINTON, 2015).

2.5 *DEEP LEARNING* EM REDES NEURAIS ARTIFICIAIS

Redes Neurais Artificiais (RNA) são inspiradas no conceito dos neurônios biológicos que compõem o cérebro humano, mais especificamente na forma como o

cérebro desempenha uma tarefa ou função (HAYKIN, 2007). Tais redes, são compostas de neurônios artificiais, organizadas em uma série de camadas conectadas. A estrutura básica de uma RNA é apresentada na Figura 7, onde os círculos verdes representam os neurônios na camada de entrada, os azuis a camada oculta e a vermelha a camada de saída (CILIMKOVIC, 2015). As setas representam as conexões entre os diferentes neurônios.

Figura 7 - Estrutura de uma RNA

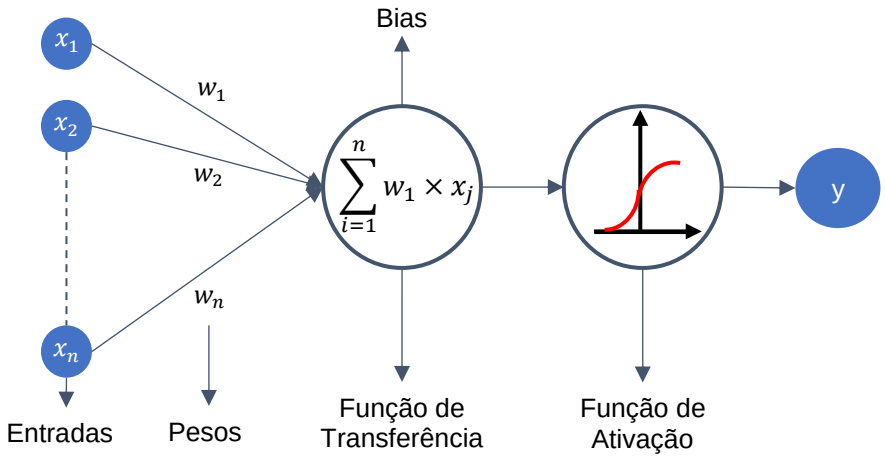


Fonte: Elaboração própria.

A camada de entrada recebe informações brutas que são alimentadas a rede. Esta parte da rede nunca muda seus valores. Cada entrada da rede é duplicada e enviada para os neurônios da camada oculta, que possuem o objetivo de modificar essas entradas usando algum valor de peso e enviar a camada de saída. Por fim, a camada de saída processa as informações recebidas da camada oculta e produz uma saída (CILIMKOVIC, 2015).

A informação flui através de uma rede neural como mostra a Figura 8. Um neurônio recebe como entrada um conjunto de valores *x* (numerados de 1 a *n*), que podem ser características de exemplos do conjunto de treino, para produzir uma saída *y*. A cada conexão de entrada, um peso *w* é associado. A função de transferência calcula a multiplicação das entradas por seus respectivos pesos mais adição de uma constante, chamada *bias*. Por fim, o resultado da função de transferência é alimentado como entrada para a função de ativação, que irá decidir quais outros neurônios devem ser acionados em seguida (GURNEY, 2014).

Figura 8 - Como a informação flui através de uma RNA



Fonte: Elaboração própria.

A função de ativação tem como objetivo limitar os dados, ou seja, mapear os dados de entrada em dados de saída em outra escala (HAYKIN, 2007). A função mais utilizada em redes neurais artificiais, e a aplicada neste trabalho, é a sigmoide. Essa função possui gráfico em formato de S, como mostra a Figura 8 e assume intervalos contínuos entre 0 e 1 (HAYKIN, 2007). Ademais, sua expressão matemática é a seguinte (NORVIG; RUSSELL, 2013):

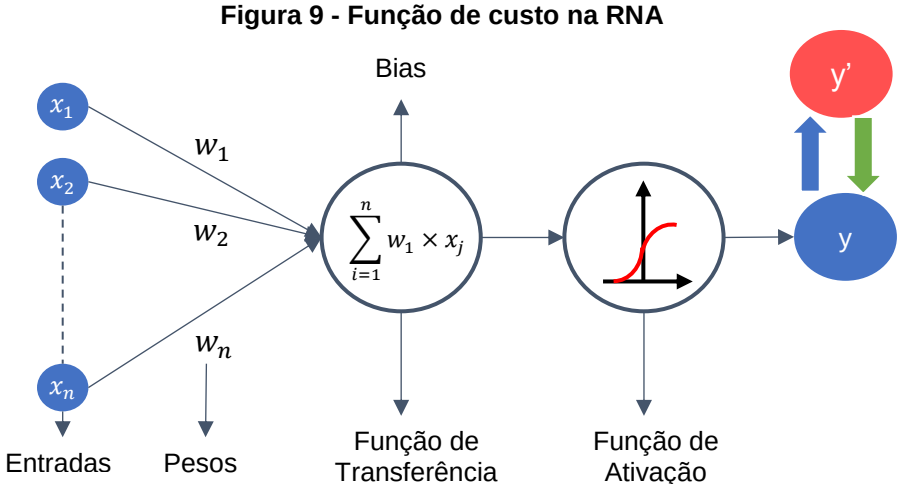
$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Além disso, uma rede neural tem a capacidade de aprender conforme os exemplos que lhes são fornecidos. Em termos gerais, o ajuste dos pesos em uma rede, normalmente é realizado através de um processo iterativo de apresentação repetida de exemplos de determinada tarefa. Em cada apresentação, pequenas modificações são feitas nos pesos para deixá-los mais alinhados com os valores desejados. A esse processo damos o nome de treinamento da rede e o conjunto de exemplos como conjunto de treinamento (GURNEY, 2014). A técnica atual mais popular para realizar o treinamento de uma rede neural, é através do algoritmo *backpropagation*, que busca a otimização dos pesos por meio da minimização da função de custo (CILIMKOVIC, 2015).

A função de custo relaciona a saída *y* fornecida pela rede, com a saída desejada *y'*, como mostra a Figura 9, ou seja, ela é responsável por mensurar o desempenho da aprendizagem para um conjunto *n* de exemplos (HAYKIN, 2007).

Neste trabalho, a função de custo empregada foi a entropia cruzada, abordada na Seção 2.5.2.

Figura 9 - Função de custo na RNA



Fonte: Elaboração própria.

O treinamento de uma rede neural por meio do aprendizado supervisionado é realizado utilizando conjuntos de dados rotulados, como mencionado anteriormente na Seção 2.4. A alimentação da rede é feita utilizando subconjuntos destes dados, chamados de lotes, pois geralmente os conjuntos de treinamento são volumosos e existe a limitação da memória computacional. Assim, quando todo o conjunto de dados é alimentado na rede, temos o que corresponde a uma época (HAYKIN, 2007).

Entretanto, um poderoso conjunto de técnicas para aprender em redes neurais e que vem sendo utilizado popularmente é conhecido como *Deep Learning* (NIELSEN, 2015). Modelos de *Deep Learning* são métodos de aprendizado compostos de múltiplas camadas de processamento, que permitem uma representação dos dados com múltiplos níveis de abstração (LECUN; BENGIO; HINTON, 2015). De forma mais simplificada, modelos de *Deep Learning* são redes neurais com múltiplas camadas por onde os dados de entrada navegam.

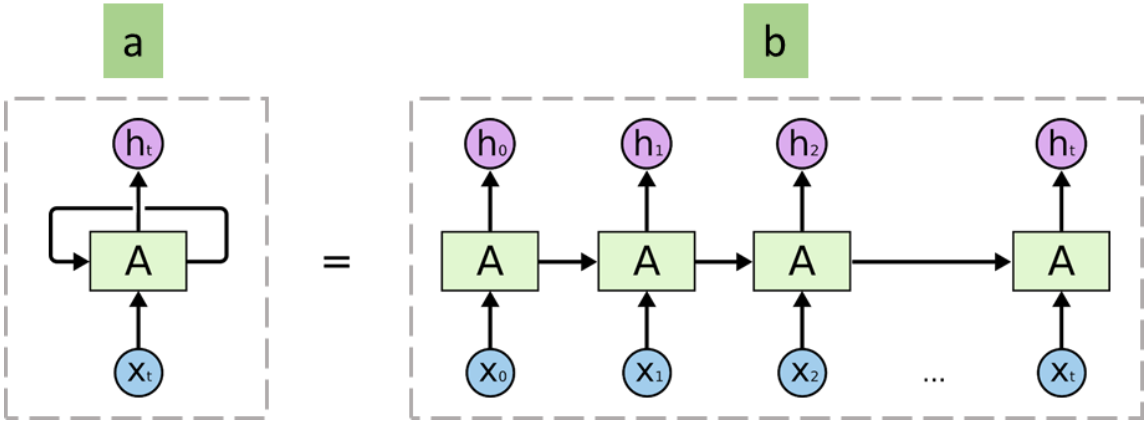
Deep Learning ganhou popularidade devido ao aumento significativo da quantidade de dados para treinamento, aos recentes avanços na área de aprendizado de máquina e às atuais habilidades de processamento dos chips, como as *Graphics Processing Units* (GPUs) (DENG; YU, 2014). O processamento paralelo das GPUs proporciona eficiência e permite que os pesquisadores explorem redes de capacidade significativamente maiores, treinando-as com volumosos conjuntos de dados

(CHETLUR et al., 2014). GPUs são usadas para acelerar o treinamento de vários tipos de redes de *Deep Learning*, como as *Recurrent Neural Networks* (RNNs) (SCHMIDHUBER, 2015), utilizadas neste trabalho.

2.5.1 *Recurrent Neural Networks*

Recurrent Neural Networks são um tipo de RNA comumente usadas para tarefas que envolvem entradas sequenciais, como fala e linguagem (SUTSKEVER; MARTENS; HINTON, 2011). Isso porque, sua arquitetura possui *loops* de realimentação da rede que permitem que a informação persista, de forma que podem processar como entrada uma sequência de elementos mantendo em suas unidades ocultas um "vetor de estado", onde está contido informações sobre a história de todos os últimos elementos da sequência (LECUN; BENGIO; HINTON, 2015).

Figura 10 - Exemplo de um típico módulo RNN



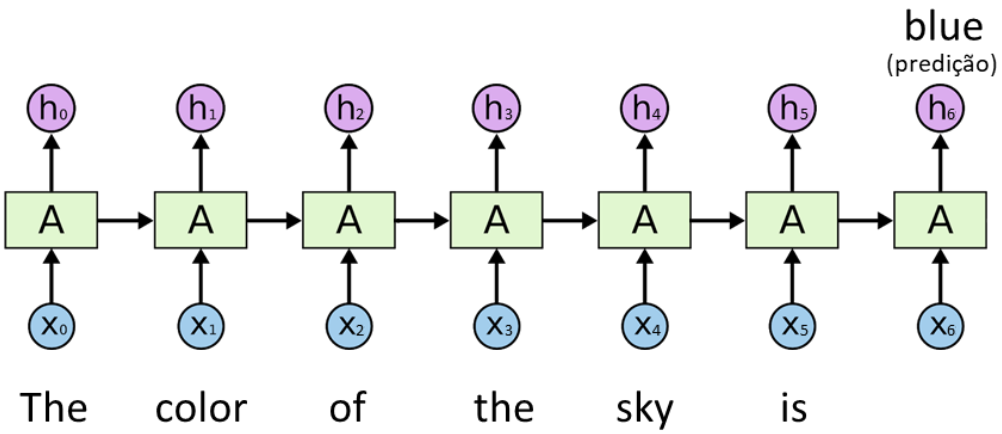
Fonte: Adaptado de Colah (2015).

A Figura 10 (a) apresenta um diagrama de blocos de um módulo RNN típico *A* que recebe uma entrada X_t e produz um valor h_t. Um *loop* permite que as informações sejam propagadas para o próximo passo. Se o *loop* do módulo *A* for “desenrolado”, como mostra a Figura 10 (b), obtemos várias cópias do mesmo bloco de rede, cada uma passando uma informação a um sucessor.

Dessa forma, enquanto uma rede neural clássica (*feedforward*) produz saídas unicamente com base em sua entrada atual, uma RNN gera saídas, considerando não

só a entrada atual, mas também baseado no histórico de entradas anteriores (NURVITADHI et al., 2016). A Figura 11, apresenta um exemplo deste último cenário, ao considerar uma RNN que prevê a próxima palavra de uma frase. Dada a frase “The color of the sky is”, a RNN calcula a probabilidade da próxima palavra e prevê “blue” (NURVITADHI et al., 2016).

Figura 11 - Exemplo de predição da próxima palavra

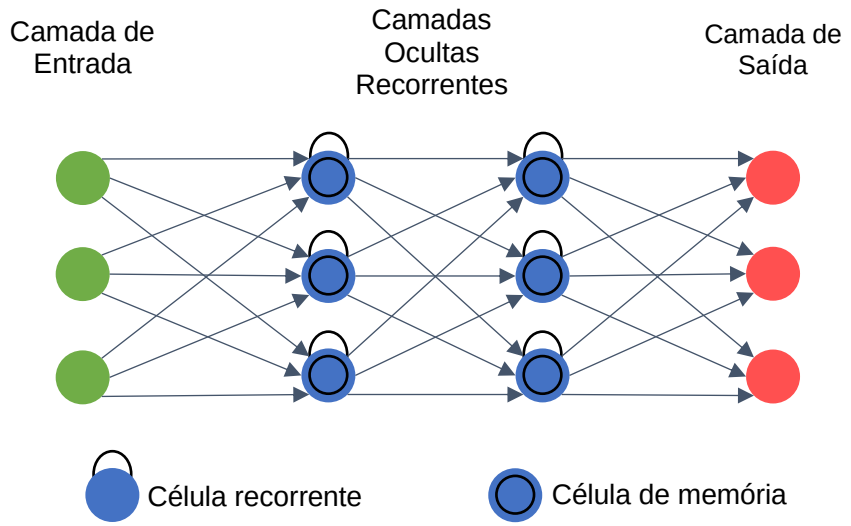


Fonte: Adaptado de Colah (2015).

Porém, apesar de as RNNs possuírem o objetivo de aprender com dependências de longo prazo (longas sequências), evidências empíricas e teóricas apontam que é difícil armazenar informações por muito tempo (LECUN; BENGIO; HINTON, 2015). No caso do exemplo da Figura 11, as informações necessárias para se prever a próxima palavra são recentes, mas considere tentar prever a última palavra no texto “*Eu cresci na França… falo fluentemente Francês*”. Informações recentes sugerem que a próxima palavra é provavelmente o nome de um idioma, no entanto, a RNN precisa armazenar informações mais antigas, como saber que o contexto é França, para prever que o idioma é Francês (NURVITADHI et al., 2016). Portanto, uma forma de corrigir isso, é aumentar a rede com uma memória de longa duração, utilizando Redes *Long Short-Term Memory* (LSTM) (LECUN; BENGIO; HINTON, 2015).

Long Short-Term Memory é uma arquitetura de RNN projetada para modelar longas sequências com mais precisão do que as RNNs convencionais (SAK; SENIOR; BEAUFAYS, 2014a). As redes LSTM são compostas de uma camada de entrada, uma ou mais células de memória que compõem as camadas ocultas recorrentes e uma camada de saída, como mostra a Figura 12 (SAK; SENIOR; BEAUFAYS, 2014b).

Figura 12 - Rede LSTM



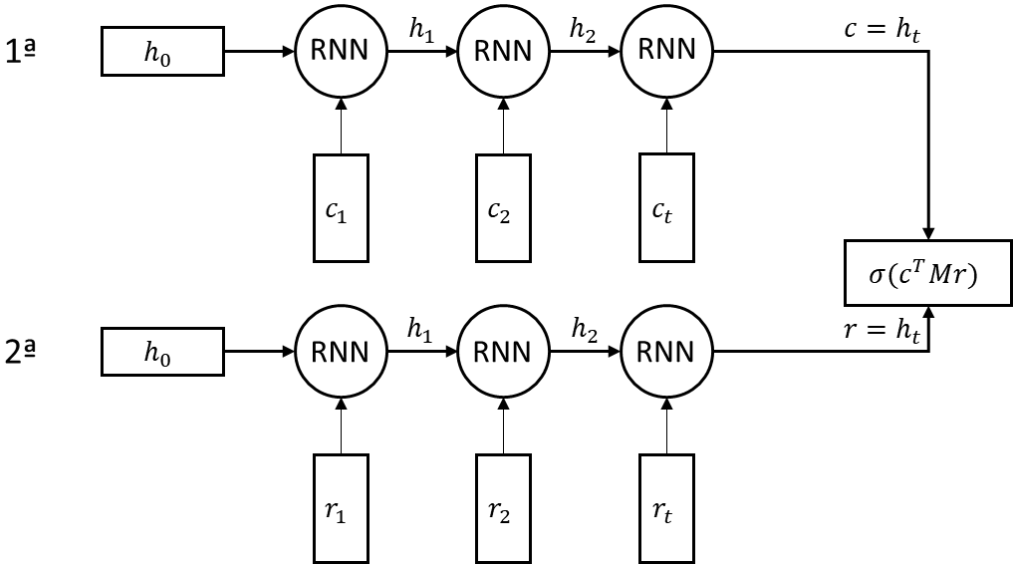
Fonte: Adaptado de Veen (2016).

Cada uma das células de memória possui três portões, que mantém e ajustam o estado da célula, sendo: um portão de esquecimento, um portão de entrada e um portão de saída. O portão de esquecimento decide quais informações são descartadas do estado da célula, o portão de entrada decide quais novas informações são armazenadas no estado da célula e o portão de saída decide quais informações do estado da célula são usadas como saída (FISCHER; KRAUSS, 2018). Considere, por exemplo, que essas células de memória funcionam como um chip de computador, onde os portões de entrada, saída e esquecimento funcionam de maneira semelhante à gravação, leitura e redefinição de sinais em um chip (ZAZO et al., 2016).

2.5.2 Modelo Dual Encoder LSTM

O modelo Dual Encoder LSTM proposto por Lowe et al. (2015), apresentado na Figura 13, faz uso de uma rede siamesa composta de duas RNNs com pesos vinculados, cujas unidades ocultas são unidades LSTM, para codificar o contexto do diálogo e uma resposta candidata. Mais precisamente, como o contexto e a resposta possuem tamanhos variados, um codificador é utilizado para se obter um vetor de tamanho fixo que os representam.

Figura 13 - Diagrama do Modelo Dual Encoder LSTM



Fonte: Adaptado de Lowe et al. (2015).

A primeira RNN da Figura 13, codifica o contexto, recebendo como entrada suas palavras representadas por um vetor de *word embeddings*, ilustrados como c_i. Da mesma forma, a segunda RNN, codifica a resposta recebendo como entrada suas *word embeddings*, denotadas por r_i.

Word embeddings são vetores de números reais, que representam palavras em um espaço n-dimensional, aprendidas a partir de grandes corpora (plural de corpus) e são capazes de captar conhecimento sintático, semântico e morfológico (HARTMANN et al., 2017). O *Word Embeddings* neste trabalho é inicializado usando os vetores pré-treinados (*Common Crawl*, 840B *tokens* de Pennington, Socher e Manning (2014)) e é ajustado durante o treinamento.

Cada vez que um *word embedding* é inserido nos codificadores de contexto ou de resposta, suas camadas ocultas são atualizadas e representações de vetores do texto inteiro são aprendidas. No final do processo de codificação, os vetores *c* e *r* ilustrados na Figura 13, representam o contexto e a resposta respectivamente como vetores de tamanho fixo.

Por fim, é calculado a probabilidade que a resposta *r* seja a resposta correta para o contexto *c*, por meio da seguinte fórmula:

$$P(r \text{ é a } resposta\ correta) = \sigma(c^T Mr) \quad (2.4)$$

Onde *M* é uma matriz de parâmetros aprendidos (pesos) e *σ* é a função sigmoide (LOWE et al., 2016). Para realizar o cálculo, podemos considerar uma abordagem generativa, de duas formas (KADLEC; SCHMID; KLEINDIENST, 2015):

a) Dado um contexto *c*, prevemos uma resposta como $r' = c^T M$ e medimos a similaridade de r' com a resposta real *r* usando produto escalar.
b) Dado uma resposta *r,* prevemos um contexto como $c' = Mr$ e medimos a similaridade de c' com o contexto real *c* usando produto escalar.

Neste trabalho foi utilizado a primeira forma (a). O resultado do produto escalar obtido é então transformado em probabilidade por meio da função sigmoide, para posteriormente calcular a função de custo.

O modelo é treinado minimizando a função de custo entropia cruzada de todos os pares rotulados (contexto, resposta) (LOWE et al., 2015, 2017). A entropia cruzada mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1. O custo da entropia cruzada aumenta à medida que a probabilidade prevista se afasta do rótulo verdadeiro.

No entanto, como a classificação do modelo é binária, isto é, os exemplos do conjunto de treino são rotulados em positivos (1) ou negativos (0), portanto a entropia cruzada pode ser calculada da seguinte forma:

$$-(y \log(p)) + (1 - y)\ \log(1 - p) \qquad (2.5)$$

Onde *y* pode assumir valor de 0 ou 1 representando o rótulo verdadeiro do exemplo de teste e *p* é a probabilidade obtida com a função sigmoide.

2.6 MÉTRICA DE AVALIAÇÃO

Para avaliar quão bom ou quão “preciso” um classificador está em prever o rótulo dos exemplos, faz-se necessário a utilização de uma métrica de avaliação amplamente usada em classificação, denominada *Recall*. Para realizar o cálculo dessa métrica são usados dois termos (HAN; PEI; KAMBER, 2012):

a) Verdadeiros Positivos (TP): Refere-se aos exemplos positivos que foram corretamente rotuladas pelo classificador.
b) Falsos Negativos (FN): São os exemplos positivos que foram erroneamente rotuladas como negativas.

Recall é uma medida de completude, isto é, refere-se a porcentagem de exemplos positivos que são rotuladas como tal. Está métrica pode ser calculada pelo total de exemplos positivos rotulados corretamente dividido pelo número total de exemplos positivos, como expressa a equação 2.6 (HAN; PEI; KAMBER, 2012):

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{2.6}$$

No entanto, como a tarefa proposta do sistema conversacional baseado em recuperação é selecionar a melhor resposta, logo é apresentado ao sistema *n* respostas candidatas ordenadas pela sua pontuação em um *ranking* para que sejam classificadas. A classificação do sistema é dada como correta se a resposta certa estiver entre as primeiras *k* candidatas classificadas (KADLEC; SCHMID; KLEINDIENST, 2015).

A métrica de avaliação utilizada em sistemas de conversação é a *Recall@k* (denotada como R@k) (KADLEC; SCHMID; KLEINDIENST, 2015; LOWE et al., 2015, 2017). Mas como o conjunto de treino usado neste trabalho é rotulado de forma binária (0,1), somente R@1 é relevante no caso da classificação binária. Entretanto, com intenção de avaliar a capacidade de recuperar um conjunto maior de respostas candidatas, neste trabalho, a avaliação foi estendida também para as métricas R@2 e R@5, assim como os trabalhos relacionados (KADLEC; SCHMID; KLEINDIENST, 2015; LOWE et al., 2015, 2017).

Portanto, durante a avaliação dos modelos, o objetivo é pontuar as respostas candidatas, de modo a se obter um vetor de pontuações ordenado de forma decrescente, constituindo um *ranking*. Por exemplo, considere um exemplo de teste, com 3 respostas candidatas para um determinado contexto, o vetor resultante durante a avaliação é composto por 3 pontuações, como [0,42, 0,30, 0,12], sendo que a resposta correta está localizada na segunda posição do vetor (0,30). Cada resposta é

pontuada de forma independente, portanto as probabilidades não precisam somarem 1.

Entretanto, o exemplo proposto seria classificado como incorreto por *Recall@1*, pois a resposta que ocupa a primeira posição do *ranking*, com pontuação de 0,42, não é a resposta correta para o contexto. No entanto, seria classificado como correto por *Recall@2*, pois a resposta correta está entre as duas primeiras posições do *ranking,* com *Recall* igual a 0,50.

3 ESTUDO DE CASO

Este capítulo descreve o estudo de caso realizado sobre um modelo de *Deep Learning* aplicado a diálogos multi-turnos baseados em recuperação. Em particular, são relatadas as bibliotecas para apoiar a tarefa de pré-processamento dos dados, construção do modelo de *Deep Learning* Dual Encoder LSTM e de um algoritmo de similaridade entre documentos, baseado no modelo de Espaço Vetorial como *baseline*. Além disso, são apresentadas as características da base de diálogos e como ela foi gerada. Por fim, os procedimentos metodológicos seguidos e os resultados obtidos da análise do impacto do contexto dos diálogos sobre a qualidade das respostas fornecidas.

3.1 BIBLIOTECAS

Todas as bibliotecas utilizadas nesta pesquisa são implementadas na linguagem de programação Python. O pré-processamento da base de dados foi feito utilizando a biblioteca de código aberto chamada *Natural Language Toolkit* (NLTK)[1] para aplicação das técnicas de tokenização, *stemming* e *lemmatization*. Para

[1] https://www.nltk.org/

manipular e analisar a base de dados, foi necessário utilizar a biblioteca Pandas[2], pois fornece ferramentas e estruturas de dados de alta performance para manipular tabelas numéricas e séries temporais.

O modelo de *Deep Learning* Dual Encoder LSTM*,* foi construído com a biblioteca de código aberto de Aprendizado de Máquina chamada TensorFlow[3], com auxílio de sua biblioteca de *Deep Learning* TF Learn[4]. Além disso, o algoritmo de similaridade entre documentos do modelo de Espaço Vetorial, usou na sua construção, algumas funções disponíveis na biblioteca de código aberto de Aprendizado de Máquina, denominada scikit-learn[5], bem como, um submódulo que reúne utilitários para construir vetores de características a partir de documentos de texto.

Para apoiar o paralelismo do treinamento do modelo Dual Encoder LSTM, a biblioteca de *Deep Learning* NVIDIA CUDA (cuDNN)[6]**,** foi utilizada por oferecer funcionalidade acelerada por GPU para operações comuns em *Deep Learning*, tendo em vista, que o TensorFlow possui abstrações apoiadas pela cuDNN.

3.2 BASE DE DIÁLOGOS

Os experimentos foram realizados utilizando o Ubuntu Dialogue Corpus[7] versão 2.0 apresentado no trabalho de Lowe et al. (2017), que consiste em um conjunto de diálogos em Inglês, extraídos a partir das conversas de bate-papo dos canais de suporte técnico do Ubuntu em uma rede pública de IRC (*Internet Relay Chat*). O Ubuntu Dialogue Corpus é um corpus de diálogo público usado como coleção de referência para pesquisas de novas soluções para sistemas conversacionais de múltiplos-turnos de conversação (KADLEC; SCHMID; KLEINDIENST, 2015; LOWE et al., 2015, 2016, 2017).

[2] https://pandas.pydata.org/
[3] https://www.tensorflow.org/
[4] http://tflearn.org/
[5] https://scikit-learn.org/
[6] https://developer.nvidia.com/cudnn
[7] Publicamente disponível em: https://github.com/rkadlec/ubuntu-ranking-dataset-creator

Originalmente, o *script* de geração do Ubuntu Corpus criado pelos autores, gera três arquivos de diálogo, que são, conjuntos para treinamento, validação e teste. Os dados de treinamento consistem em 1.000.000 de exemplos (ou instâncias), dentre os quais 50% são exemplos positivos (*label* 1) e 50% são negativos (*label* 0). Cada exemplo consiste em um *context* (contexto) referente a conversa até um certo ponto do diálogo e uma *utterance* (fala ou frase), que é a próxima fala dada como resposta ao *context*. Um rótulo positivo significa que uma *utterance* foi dada como uma resposta real (isto é, que de fato é esperada como próxima fala) a um *context*, e um rótulo zero significa que a *utterance* não é uma resposta real – mas foi escolhida aleatoriamente entre outras *utterances*, exceto a verdadeira, do *corpus*. O Quadro 2 contém alguns exemplos do conjunto de treino.

Quadro 2 - Exemplos do conjunto de treino.

Context	Utterance	Label
i 'm not suggest all - onli the one you modifi . __eou__ __eot__ ok , it sound like you re agre with me , then __eou__ though rather than the one we modifi, my idea be the one we need to merg __eou__ __eot__	sorri __eou__ i think it be ubuntu relat . __eou__	0
i think d-i instal the relev kernel for your machin . i have a p4 and it instal the 386 kernel __eou__ holi crap a lot of stuff get instal by default :) __eou__ you be instal vim on a box of mine __eou__ ;) __eou__ __eot__	that the one __eou__	1

Fonte: Elaboração própria.

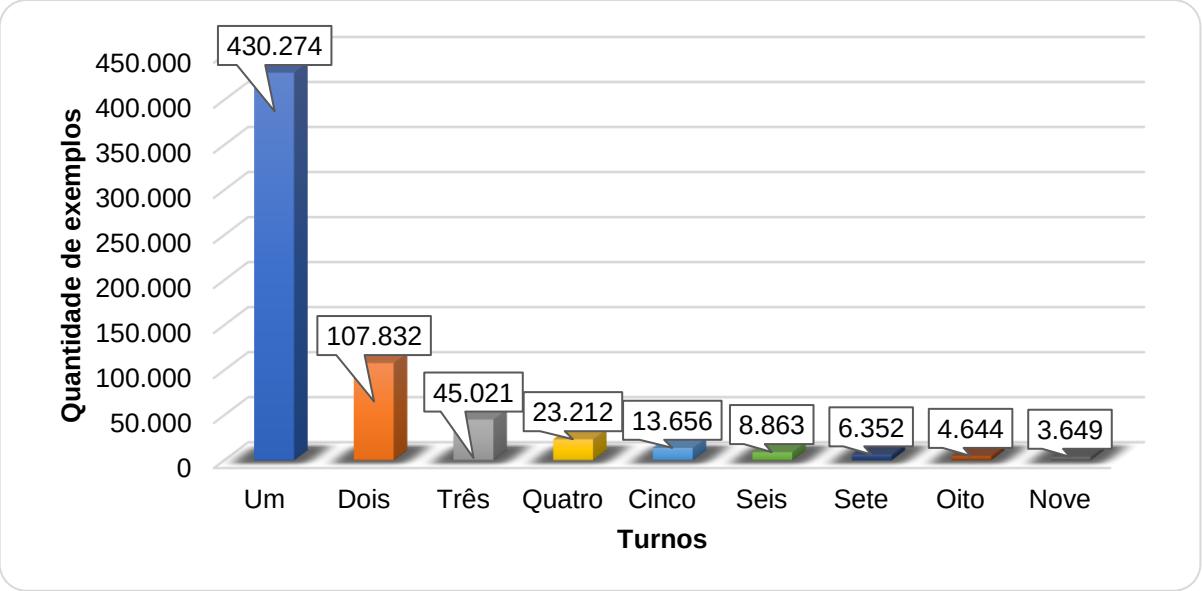
É possível perceber, que existem *tags* que fazem a diferenciação entre o final de uma *utterance* (__eou__) e o final de um turno (__eot__). Essa terminologia é consistente entre o conjunto de treinamento, teste e validação.

O formato dos conjuntos de teste e validação é diferente do conjunto de treinamento. Cada exemplo no conjunto de teste/validação consiste em um *context*, uma *utterance* que representa a resposta esperada e verdadeira, e nove *utterances* incorretas chamadas de distratores. Assim, o objetivo do modelo de aprendizado é atribuir uma pontuação (*score*) mais alta a *utterance* verdadeira e pontuações mais baixas a *utterances* incorretas, por isso a utilização de distratores.

No entanto, o conjunto de treinamento gerado com o *script* original é constituído de *contexts* de tamanhos aleatórios, portanto, após analisá-lo, foi possível observar

que os *contexts* deste conjunto são compostos de um a nove turnos de interação. A Figura 14, mostra a quantidade de exemplos por turnos no conjunto de treinamento. Vale ressaltar, que os conjuntos de teste e validação também são constituídos de *contexts* de tamanhos aleatórios.

Figura 14 - Quantidade de exemplos por turnos no conjunto de treino aleatório



Fonte: Elaboração própria.

Lowe et al. (2017) determinou o tamanho do *context* de forma estocástica, por amostragem uniforme utilizando a seguinte equação:

$$c = \sim Unif(2, t-1) \qquad (3.1)$$

Na equação 3.1, *c* denota o tamanho do *context* e o parâmetro *t* é o tamanho real do diálogo, logo existe uma restrição que $c \leq t - 1$. Na prática, isso leva a diálogos de teste curtos com *contexts* curtos, enquanto diálogos mais longos são frequentemente divididos em uma combinação de *context* curto, médio e longo, sendo que os *contexts* mais longos são considerados mais desejáveis (LOWE et al., 2017).

Diferente de Lowe et al. (2015, 2017), nesta pesquisa, o estudo realizado tinha como objetivo entender o impacto do tamanho do *context* na previsão da próxima *utterance*. Isto é, entender se a escolha de exemplos com *contexts* compostos de 2, 3, 4 ou mais turnos como conjunto de treinamento, teria algum impacto na previsão das próximas *utterances*.

Como o trabalho de Lowe et al. (2015, 2017) detalha a criação do *corpus* e disponibiliza o *script* de geração da base de diálogos no Github, foi possível a modificação do mesmo, para obter conjuntos de treinamento com a quantidade de turnos desejada.

3.2.1 Modificação do *Script* de Geração do Conjunto de Treino

O script original de geração do Ubuntu Corpus é composto de funções responsáveis por extrair diálogos entre dois agentes e fazer a amostragem aleatória de todos os conjuntos de diálogos (treino, validação e teste) com exemplos positivos e negativos.

Esse *script* recebe como argumento o diretório onde os diálogos estão localizados, a semente para gerar números aleatórios (definida por padrão como sendo igual a 1234), o nome do arquivo de saída para gravação em formato CSV e as *flags*, que representam as técnicas de PLN, tokenização (-t), *stemming* (-s) e *lemmatization* (-l), para pré-processamento dos dados.

O *script,* ainda permite passar sub argumentos para geração de cada um dos tipos de conjuntos de diálogos, bem como, modificar a probabilidade da seleção de exemplos positivos, que é calculada a partir da proporção de exemplos positivos para o total de exemplos no conjunto de treinamento (definido por padrão como sendo igual 0,5). Além disso, é possível determinar o número de exemplos de treinamento que deve ser gerado (definido por padrão como sendo igual a 1.000.000) e o número de exemplos distratores para cada *context* (definido por padrão como sendo igual 9) nos conjuntos de teste e validação.

Nesta pesquisa foi considerado somente como o conjunto de treinamento foi gerado, uma vez que foram mantidos os conjuntos de teste e validação criados com o *script* original. Portanto, o *script* original utiliza 8 funções para gerar o conjunto de treino, das quais, 3 delas foram unificadas para criar exemplos de treino com base no número de turnos desejado, além disso, foi adicionado mais um sub-argumento (-t) que possibilita ao usuário passar a quantidade de turnos por argumento.

A nova função se chama “create_examples_train”[8], ela é a unificação das funções “create_examples”, “create_single_dialog_train_example” e “create _random_context”, com adição de algumas modificações. Em linhas gerais, o que cada uma dessas funções realiza no *script* original é o seguinte:

a) create_examples: Cria uma lista de exemplos de treinamento a partir de uma lista de diálogos e funções que transforma um diálogo em um exemplo.
b) create_single_dialog_train_example: Cria um único exemplo para o conjunto de treinamento.
c) create_random_context: Cria um *context* com tamanho determinado de forma estocástica.

Basicamente, a função “create_examples_train” entra em um laço de repetição, cujo número de iterações é determinado pela quantidade de exemplos informada. A cada iteração, é realizada uma busca no diretório de diálogos, com objetivo de encontrar um diálogo que seja maior ou igual ao tamanho de turnos desejado, como mostra o trecho de código da Figura 15. Assim que um diálogo é encontrado, o mesmo será processado de modo a obter a tripla (*context*, *utterance*, *label*), para compor um exemplo, que por fim será adicionado a uma lista de exemplos.

Figura 15 - Trecho de código da função "create_examples_train".

Trecho de Código Python

```
1   while i < examples_num:
2       while True:
3           context_dialog = candidate_dialog_paths[x % unique_dialogs_num]
4           dialogOriginal = translate_dialog_to_lists(context_dialog)
5           max_len = min(max_context_length, len(dialogOriginal)) - 2
6           x += 1
7       if max_len >= num:
8           Break
9   context_turns = rng.randint(num,max_len)
10  dialog = dialogOriginal[:context_turns]
11  dialog = dialog[-num:]
```

Fonte: Elaboração própria.

[8] Disponível em: https://github.com/amycardoso/retrieval-based-chatbot/

Para melhor compreensão, considere a criação de um exemplo de treino com contexto de tamanho 2, representado no Quadro 4, a partir do diálogo ilustrado no Quadro 3. Note que as *utterances* do Quadro 3 de um mesmo usuário são colocadas em sequência cronológica em uma lista de *utterances*, num total de quatro listas conforme ilustrada no Quadro 4. As quatro listas compõem os dois turnos do contexto da conversa, sendo que cada turno é uma entrada de texto formada por uma ou mais *utterances*, acompanhada de uma resposta composta por uma ou mais *utterances*.

Quadro 3 - Exemplo de diálogo

Usuários	**Utterances**
seb128	hum, just mail me if you want
seb128	I'll follow the mail on the internal list
seb128	Yes
bob2	gnome-settings-daemon is working!
seb128	Kamion has fixed it
bob2	everything still has .menu files, right? so installing the menu package and some hackery can get the Debian menu back?

Fonte: Elaboração própria.

Quadro 4 - Lista de *utterances* para compor contexto de tamanho dois a partir do diálogo de exemplo

1	['hum, just mail me if you want', "I'll follow the mail on the internal list", 'yes']
2	[‘gnome-settings-daemon is working!']
3	['Kamion has fixed it']
4	['everything still has .menu files, right? so installing the menu package and some hackery can get the Debian menu back?']

Fonte: Elaboração própria.

Para evitar que existam *contexts* similares, na linha 9 do trecho de código (Figura 15), um número inteiro aleatório *N* é gerado no intervalo *a* <= *N* <= *b*, onde *a* é o número de turnos desejado, e *b* é o tamanho máximo que um *context* pode ter. Assim, o diálogo é cortado a partir da posição aleatória sorteada, por esse motivo, até diálogos que sejam maiores que a quantidade de turnos, são selecionados pela função, uma vez que serão cortados, o que diminui o tempo de busca no diretório.

Assumindo que o número aleatório sorteado é 3, o diálogo será cortado a partir da posição 3, resultando no Quadro 5. Mas como desejamos criar um *context* com um turno, selecionamos as duas últimas posições da lista de *utterances*, e a próxima *utterance* da lista será usada como resposta ao *context*, isto é, se esse exemplo for rotulado como positivo, se não, uma *utterance* qualquer dentro do *corpus* será usada como resposta e rotulada como exemplo negativo.

Quadro 5 - Diálogo cortado a partir da posição selecionada

1	['hum, just mail me if you want', "I'll follow the mail on the internal list", 'yes']
2	[‘gnome-settings-daemon is working!']
3	['Kamion has fixed it']

Fonte: Elaboração própria.

A lógica para rotulação de exemplos do *script* original foi mantida sem alterações no *script* modificado. Assim, um número aleatório é gerado e se ele for menor que a probabilidade (considere 0,5), logo o exemplo recebe rótulo 1, se não, recebe rótulo 0. No caso do exemplo proposto no Quadro 6, ele receberá rótulo 1, pois foi assumido que o número aleatório gerado é menor que 0,5. Portanto, o exemplo criado com a função “create_examples_train” é mostrado no Quadro 6.

Quadro 6 - Exemplo gerado com a nova função

Context	Utterrance	Label
hum, just mail me if you want __eou__ I\'ll follow the mail on the internal list __eou__ yes __eou__ __eot__ gnome-settings-daemon is working! __eou__ __eot__	Kamion has fixed it __eou__ __eot__	1

Fonte: Elaboração própria.

Dessa maneira, a partir do *script* modificado, foram gerados 9 conjuntos de treino distintos, onde o conjunto 1 é formado por exemplos com *contexts* de tamanho 1, conjunto 2 formado por exemplos com *contexts* de tamanho 2 e assim por diante, até 9. Cada conjunto contem 1.000.000 de exemplos, dos quais 50% destes exemplos são positivos e os outros 50% são negativos. Além disso, utilizando o *script* original, foram gerados os conjuntos de teste, validação e mais um de treino formado por exemplos com *contexts* de tamanho aleatório para efeito de comparação com os demais 9 conjuntos de treino, totalizando 10 conjuntos de treino, 1 de teste e 1 de validação.

3.3 METODOLOGIA DE EXPERIMENTAÇÃO

A metodologia de experimentação empregada foi composta de quatro etapas: (i) pré-processamento dos dados; (ii) aplicação dos modelos Dual Encoder LSTM e Espaço Vetorial; (iii) avaliação dos modelos; (iv) e predição de respostas dadas pelos modelos. Esse processo foi repetido 10 vezes, devido aos 10 conjuntos de treinamento, sendo que os conjuntos de teste e validação foram os mesmos para todas as repetições.

Na etapa de pré-processamento, além da aplicação de técnicas de PLN, foi necessário converter os conjuntos de diálogos para o formato recomendado do TensorFlow (TFRecord), visto que originalmente possuíam formato CSV. Também foram criados vocabulários que mapeiam cada palavra para um número inteiro, por exemplo "Chatbot" pode se tornar 1728. Assim, os arquivos TFRecord gerados armazenam esses números inteiros em vez de *strings*, e é por meio do vocabulário que os inteiros são mapeados para palavras.

Quadro 7 - Configurações do ambiente

Processador	Intel® Core™ i5-8250U CPU @ 1.60Hz 1.80 GHz
Memória RAM	8,00 GB (utilizável: 7,88 GB)
Sistema Operacional	Ubuntu 18.04 64 bits
Placa de Vídeo	NVIDIA® GeForce® MX150 com 2GB dedicados GDDR5
Disco Rígido	1 TB

Fonte: Elaboração própria.

Na etapa de aplicação dos modelos, foram realizados testes iniciais com o modelo Dual Encoder LSTM em um ambiente com as configurações apresentadas no Quadro 7. Mas, por conta do grande volume de dados utilizado para treinamento, ficou inviável concluir tais testes pois houve consumo de toda a memória RAM disponível e pouco aproveitamento do paralelismo da CPU no treinamento do modelo.

Dessa forma, fez-se necessário a utilização de CUDA[9] (*Compute Unified Device Architecture*) para apoiar o paralelismo do treinamento do modelo Dual

[9] https://developer.nvidia.com/cuda-zone

Encoder LSTM. A CUDA, é uma plataforma de computação paralela e modelo de programação desenvolvido pela NVIDIA, que proporciona um aumento drástico na performance de aplicativos aproveitando a potência das GPUs. Assim, utilizando CUDA o processo de treinamento do modelo Dual Encoder LSTM que antes durava 24 horas, passou a durar, em média, 2 horas.

O treinamento do modelo Dual Encoder LSTM consistia em vários passos, que se refere ao número de vezes que o *loop* de treinamento do modelo é executado para atualizar seus parâmetros internos. Em cada passo do *loop*, o modelo processava um lote de dados. O tamanho do lote de dados foi definido com tamanho 64, isto é, a cada passo, o modelo era alimentado com 64 exemplos do conjunto de treinamento e a cada 2000 passos, seus parâmetros eram ajustados com utilização do conjunto de validação. Além disso, foi implementada uma função para interromper seu treinamento, assim que o valor de *Recall@1* (R@1) não aumentar mais sobre o conjunto de validação depois de 4000 passos, como mostra a Figura 16.

Figura 16 - Função para interromper treinamento

Trecho de Código Python

```
1  eval_monitor = tf.contrib.learn.monitors.ValidationMonitor(
2          input_fn=input_fn_eval,
3          every_n_steps=FLAGS.eval_every,
4          metrics=eval_metrics,
5          early_stopping_metric="recall_at_1",
6          early_stopping_metric_minimize=False,
7          early_stopping_rounds = 4000)
```

Fonte: Elaboração própria.

Na etapa de avaliação, para cada conjunto de treinamento avaliado, foi obtido o valor da métrica *Recall@k* (denotada como R@k) sobre o conjunto de teste, sendo que o sistema acertava quando a resposta correta estava entre as primeiras *k* respostas candidatas do *ranking*. Por fim, a última etapa consistiu em realizar predições para obter pontuações de respostas dadas a *contexts* que não foram vistos antes pelos modelos.

3.4 EXPERIMENTOS E RESULTADOS

Nesta seção são reportados os resultados de dois experimentos que possuíam objetivos distintos. A seção 3.4.1 aborda os resultados do impacto dos modelos Dual Encoder LSTM e Espaço Vetorial sobre conjuntos aleatórios e variáveis de turnos de interação. Enquanto a seção 3.4.2 discute os resultados do impacto do número de exemplos no treinamento do modelo Dual Encoder LSTM.

No entanto, é importante frisar que o modelo de Espaço Vetorial não envolve aprendizado, portanto, para efeito de comparação, os mesmos conjuntos de diálogos, que aqui denotamos como conjuntos de treinamento, foram usados para calcular os vetores TF-IDF de cada exemplo e para treinamento do modelo que efetivamente usa aprendizado de máquina, o Dual Encoder LSTM. Além disso, o mesmo conjunto de teste foi utilizado para ambos modelos e experimentos.

3.4.1 Resultados da análise do impacto do contexto

Os resultados alcançados após aplicação dos modelos Dual Encoder LSTM e Espaço Vetorial sobre os 10 conjuntos de diálogos podem ser observados nas Tabelas 1 e 2, respectivamente. Em relação ao desempenho individual, o modelo Dual Encoder LSTM, obteve melhores resultados em R@1, R@2 e R@5 quando treinado com exemplos compostos por *contexts* de tamanhos aleatórios. Enquanto que o modelo de Espaço Vetorial, obteve melhores resultados quando este modela os contextos dos diálogos formados unicamente por exemplos de dois, três, até nove turnos.

É possível observar na Tabela 1, o decaimento do desempenho do LSTM, a medida que este é treinado com exemplos formados unicamente por contextos maiores que dois turnos. Este decaimento foi mais acentuado considerando a melhor resposta do *ranking* de candidatas no LSTM. Em contrapartida, o modelo Vetorial foi

menos suscetível a variação dos exemplos usados na construção dos vetores de documentos no espaço vetorial.

Tabela 1 - Resultados do Modelo Dual Encoder LSTM

		Turnos								
Recall	**Aleatório**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
R@1	0,528	0,481	0,475	0,466	0,442	0,409	0,361	0,325	0,288	0,177
R@2	0,706	0,655	0,648	0,645	0,619	0,591	0,533	0,495	0,447	0,311
R@5	0,912	0,885	0,888	0,885	0,875	0,856	0,816	0,790	0,747	0,619

Fonte: Elaboração própria.

Tabela 2 - Resultados do Modelo de Espaço Vetorial

		Turnos								
Recall	**Aleatório**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
R@1	0,494	0,490	0,496	0,499	0,500	0,500	0,500	0,499	0,497	0,496
R@2	0,596	0,592	0,597	0,600	0,602	0,602	0,602	0,602	0,601	0,599
R@5	0,766	0,760	0,766	0,773	0,775	0,776	0,776	0,777	0,777	0,776

Fonte: Elaboração própria.

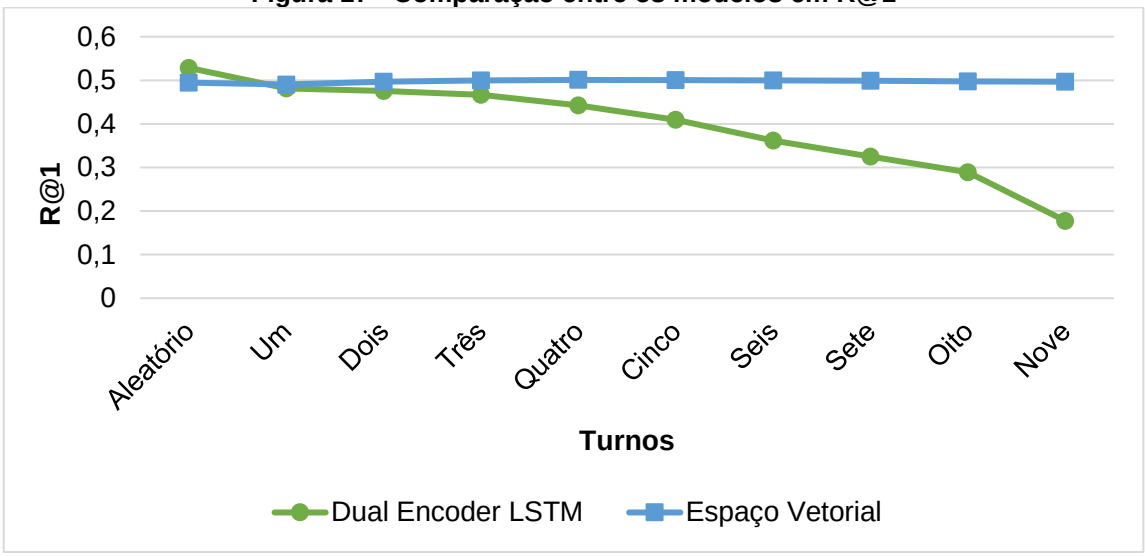
Para ilustrar essa situação, tome como referência os resultados dos modelos LSTM e Vetorial para o conjunto aleatório e observe que o efeito desta variação de tamanhos de contexto em R@1, tornou o LSTM pior que o modelo Vetorial. LSTM chegou a ser aproximadamente 9% pior quando treinado somente com exemplos de contexto com um turno de interação. Essa piora aumenta com a variação e chega a ser, no caso extremo, de aproximadamente 66% quando treinado com exemplos de nove turnos de interação. Para o modelo Vetorial, a variação tornou este modelo somente 1% pior em todos os casos.

Por outro lado, se o interesse é obter um *ranking* com mais respostas candidatas, observou-se que para R@2 e R@5, mesmo que o LSTM tenha seu desempenho ainda muito afetado pela variação do treino, ainda assim, consegue ser superior ao modelo Vetorial. Novamente, tomando como referência os resultados de ambos modelos usando o conjunto de treino aleatório, mas agora para R@2, notou-

se que LSTM se saiu melhor nos casos onde o contexto tem tamanho de um, dois e três turnos. Enquanto que, de cinco a nove turnos, os resultados do LSTM demonstram uma piora no desempenho em relação ao Vetorial. Mas, para a métrica R@5, LSTM se saiu melhor com conjuntos formados unicamente por diálogos com contexto de um até sete turnos de interação.

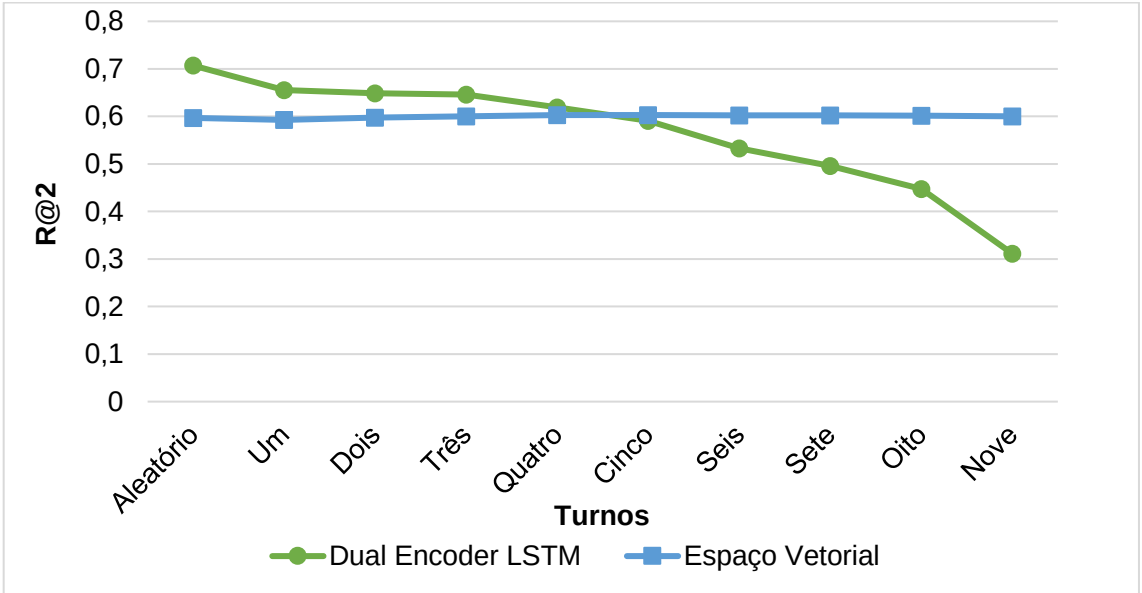
As Figuras 17 e 18 são gráficos que ilustram o efeito da variação do tamanho do contexto, considerando os resultados obtidos pelos modelos para R@1 e R@2, descritos nas Tabela 1 e 2. O eixo x representa a quantidade de turnos no contexto e o eixo y representa o valor da métrica.

Figura 17 - Comparação entre os modelos em R@1



Fonte: Elaboração própria.

Figura 18 - Comparação entre os modelos em R@2

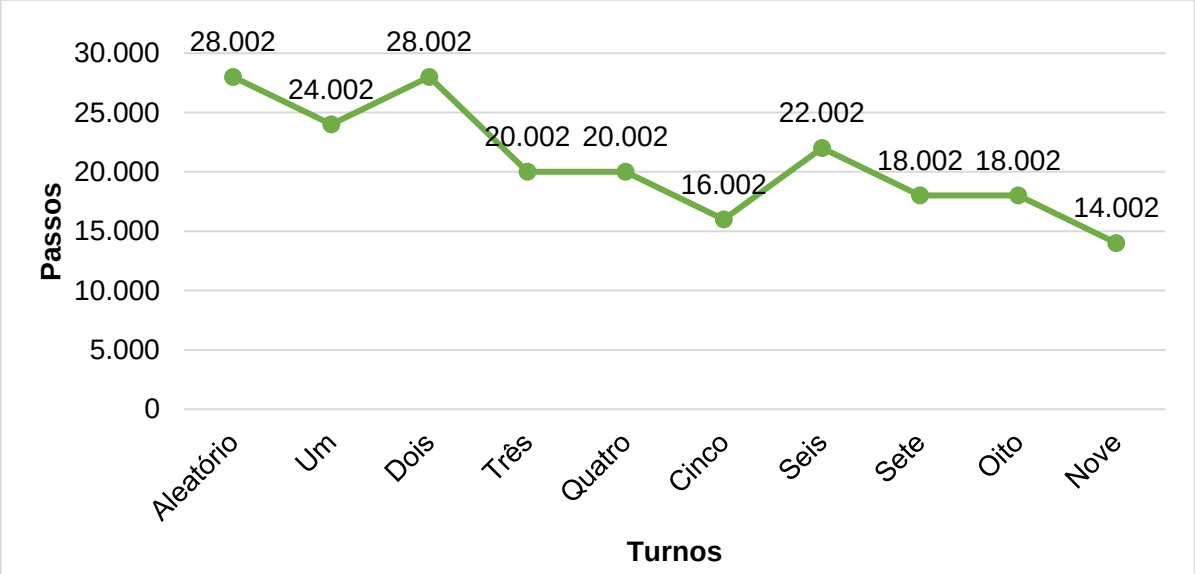


Fonte: Elaboração própria.

De acordo com o que foi evidenciado nas Figuras 17 e 18, é possível perceber que o desempenho do modelo LSTM tem um declínio a medida que a quantidade de turnos no contexto aumenta. Ainda que para R@2, seu desempenho seja superior ao Vetorial até o conjunto de três turnos, logo ele sofre um decaimento de quatro a nove turnos. Enquanto que o modelo Vetorial mantém comportamento estável para R@1 e R@2.

Além disso, sabendo que durante o treinamento o modelo Dual Encoder LSTM recebia 64 exemplos por passo, a quantidade de passos necessárias para seu treinamento diminuiu quando o tamanho do contexto aumentava gradativamente, isto é, o modelo precisava de menos exemplos para treinamento, como mostra a Figura 19. No eixo y a escala é graduada de 5 mil em 5 mil passos.

Figura 19 - Quantidade de passos por conjunto de treinamento



Fonte: Elaboração própria.

Vale ressaltar, que na Figura 19, apenas o conjunto de treinamento de dois turnos obteve a mesma quantidade de passos que o conjunto aleatório, igual a 28.002 passos, enquanto que ao utilizar um conjunto com contextos maiores, como de cinco turnos e de nove turnos, foram necessários apenas 16.002 e 14.002 passos respectivamente.

Também foi selecionado de forma aleatória um exemplo de predição de respostas com o modelo Dual Encoder LSTM. A Figura 20 mostra um contexto composto de um turno e suas respectivas respostas candidatas. Nesse exemplo, o

modelo treinado com o conjunto aleatório, obteve maior pontuação igual a 0.90, a resposta correta, que está destacada em negrito.

Figura 20 - Exemplo de predição de resposta

Context:	**User A:** hey guy , how can i add an extra xsession to ubuntu 10.04 ? __eou__ **User A:** that be , i dont want gnome __eou__ **User B:** tri this : https : //wiki.ubuntu.com/customxsess __eou__
Pontuação	**Respostas Candidatas**
0.90	**ill take a look , thank __eou__**
0.80	.. thank you i ll check those out as well . __eou__
0.60	most mainboard boot from sata port1 to sata port x __eou__ master /slave be ide __eou__
0.30	ok , i ll ignor the page for now __eou__
0.33	answer my question __eou__
0.06	when i come here , it complet irrit me peopl have to give me botlink . i liek when peopl can just tell me without all the extra work . __eou__ this be n't life , it 's an oper system . do n't give me that ! __eou__
0.06	now the same partit , but free space on that disk . i cancel the instal befor instal the boot , so at the moment i can perfect boot w7 , but now linux __eou__
0.02	that 's just the chroot shell . `` exit '' will close the shell . __eou__
0.02	what doe `` restart my pc not visibl anoth hdd '' mean ? __eou__ ls __eou__
0.0004	;) i 'm not surpris . good luck . __eou__ pcmanfm be solid i hear , but reli on the udev mount stuff (that you be current look for !) so it 's a littl heavi instal wise for my like unfortuneat : (__eou__ nice find though ! __eou__

Fonte: Elaboração própria.

3.4.2 Variando a quantidade de dados de treinamento

Com o objetivo de observar como a quantidade de exemplos pode afetar os resultados do modelo Dual Encoder LSTM, foram gerados conjuntos que variam de 10^2 á 10^5 de exemplos. Além disso, como critério do tamanho dos contextos que constituíram os exemplos desses conjuntos, foi levado em consideração os dois melhores resultados obtidos com o modelo em questão, apresentados anteriormente

na Tabela 1. Dessa forma, foram gerados 4 conjuntos compostos de contextos de tamanhos aleatórios e 4 conjuntos compostos de contextos de um turno, cujos resultados após treinamento do modelo, são apresentados nas Tabelas 3 e 4 respectivamente, com adição dos valores obtidos e apresentados anteriormente (Tabela 1) com 10^6 exemplos.

Tabela 3 - Resultados variando a quantidade de exemplos no treino aleatório

	10^2	10^3	10^4	10^5	10^6
R@1	0,301215	0,397780	0,347674	0,250317	0,528541
R@2	0,340380	0,425422	0,385887	0,407716	0,706976
R@5	0,536892	0,588107	0,591543	0,712737	0,912949

Fonte: Elaboração própria.

Tabela 4 - Resultados variando a quantidade de exemplos no treino de um turno

	10^2	10^3	10^4	10^5	10^6
R@1	0,291067	0,344556	0,342494	0,250898	0,481501
R@2	0,314904	0,376109	0,378276	0,411997	0,655602
R@5	0,536892	0,588107	0,591543	0,712737	0,885676

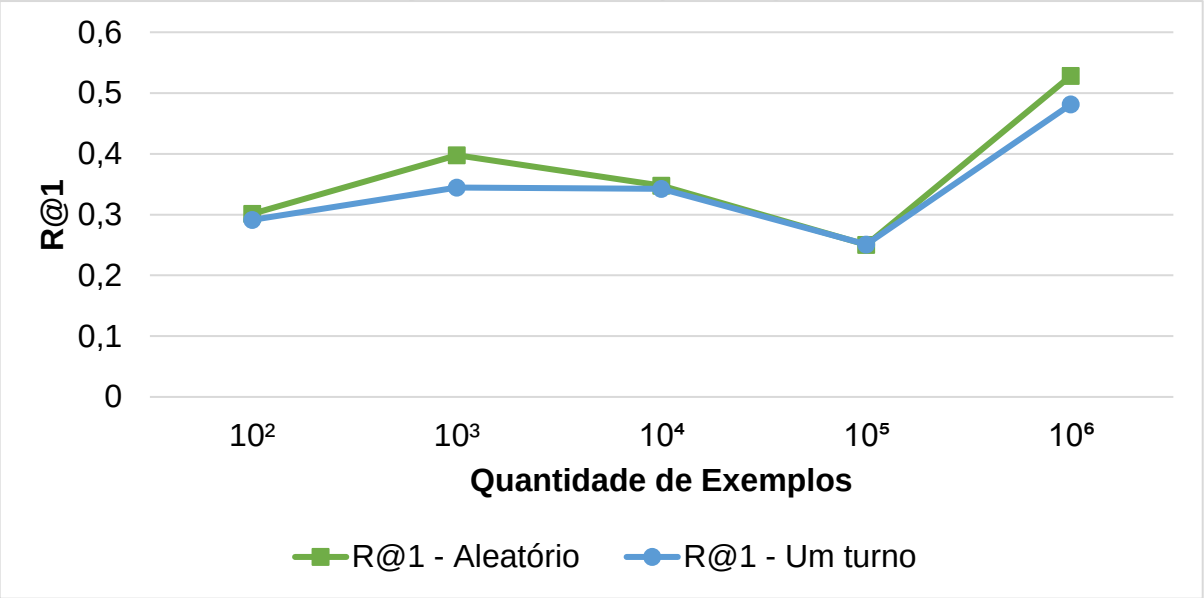
Fonte: Elaboração própria.

De acordo com os resultados, constatou-se que a quantidade de exemplos no treino influencia no desempenho do modelo, pois sua performance é relativamente inferior quando treinado com poucos dados. No caso da utilização de contextos de tamanhos aleatórios, o modelo treinado com 10^3 exemplos reduziu o valor de R@1 para 24,81% em relação a 10^6 exemplos. Enquanto que utilizando contextos de um turno, o modelo também treinado com 10^3 exemplos, reduziu o valor de R@1 em 28,48% em relação a 10^6 exemplos.

Além disso, ao treinar o modelo com os 8 novos conjuntos de exemplos, foi possível obter sua curva de aprendizagem a partir da Tabela 3 e 4, como mostra a Figura 21. O eixo x representa a quantidade de exemplos nos conjuntos de

treinamento e o eixo y representa o valor de R@1. Percebe-se que a diferença no treinamento do modelo com contextos aleatórios ou de um turno é mínima, porém é visível que quando treinado com grandes volumes de dados, o gráfico apresenta um salto entre 10^5 e 10^6, pois o modelo aumenta seu aprendizado em 111,2% quando treinado com contextos aleatórios e em 92,4% com contextos de um turno.

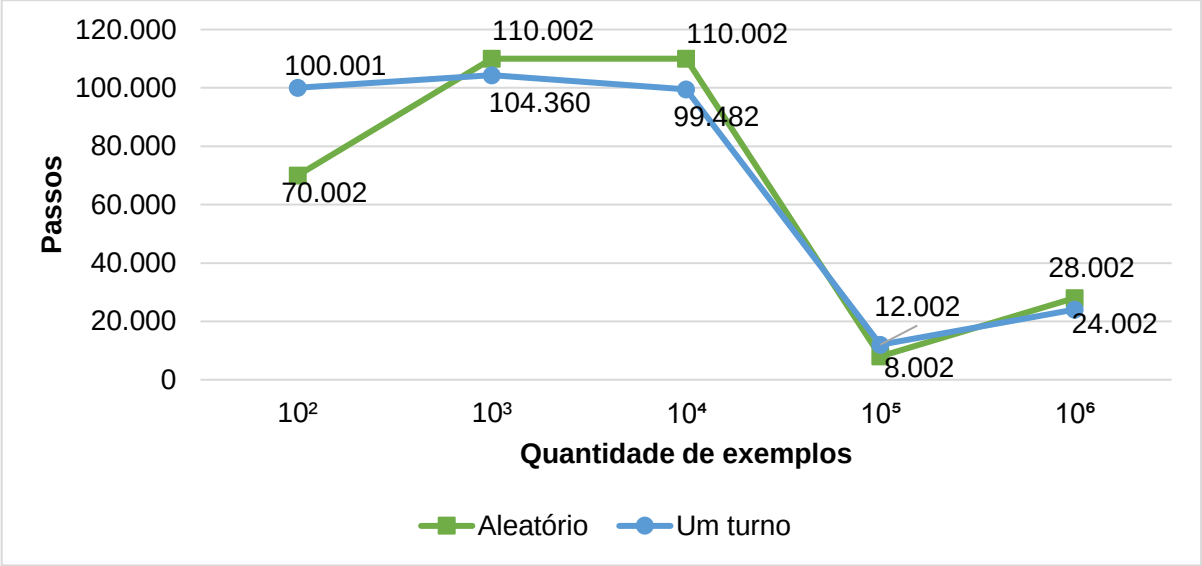
Figura 21 - Curva de aprendizagem



Fonte: Elaboração própria.

Por outro lado, a quantidade de passos necessários para treinamento é bem maior quando são usados menos dados, chegando a ser necessário mais de 100.000 passos, como mostra a Figura 22.

Figura 22 - Quantidade de passos com menos dados



Fonte: Elaboração própria.

Diante do que foi exposto nas Figuras 21 e 22, é possível concluir que o modelo Dual Encoder LSTM necessita de um grande volume de dados para treinamento, como 10^6 de exemplos. Tendo em vista que, usando menos dados, seu desempenho é reduzido e a tarefa de treinamento torna-se dispendiosa.

4 CONSIDERAÇÕES FINAIS E RECOMENDAÇÕES

Neste capítulo são apresentadas as considerações finais relacionadas a este trabalho, bem como as recomendações para trabalhos futuros.

4.2 CONSIDERAÇÕES FINAIS

O objetivo dessa pesquisa foi construir os modelos Dual Encoder LSTM e Espaço Vetorial, e avaliar o impacto do tamanho do contexto de conversação sobre a qualidade das respostas fornecidas por eles.

Para cumprir esse objetivo, primeiramente, foi feita uma revisão dos conceitos que dão embasamento a esta pesquisa, e partir disso, selecionar uma base de diálogos comumente utilizada em pesquisas como esta. Além disso, modificações foram realizadas no *script* de geração, mais especificamente, na geração de conjuntos de treinamento, para que os contextos que constituíssem os exemplos desses conjuntos, possuíssem a quantidade de turnos de interação que fosse desejada. Também foram realizadas adaptações nas implementações dos modelos. Em seguida foram realizados experimentos, aplicando os modelos sobre os conjuntos gerados e os resultados foram observados e discutidos.

Com base no exposto, conclui-se que a pesquisa atingiu o seu objetivo, uma vez que ambos modelos foram construídos, avaliados e comparados, evidenciando o impacto do tamanho do contexto.

Tendo em vista que, o modelo de *Deep Learning* Dual Encoder LSTM é projetado para lidar com contextos longos, pois faz uso de unidades LSTM, ele obteve desempenho reduzido à medida que o tamanho do contexto aumentava, chegando a ser aproximadamente 66% pior quando treinado com exemplos de nove turnos de interação. Enquanto que o modelo de Espaço Vetorial, utilizado como *baseline,* realizou cálculos de similaridade entre o contexto e a resposta candidata, sem aprendizado de máquina e obteve desempenho estável, ou seja, foi menos suscetível a variação do tamanho do contexto.

Na variação de turnos, o modelo Dual Encoder LSTM obteve melhor resultado para R@1, com um turno de interação, portanto, sistemas conversacionais que trabalham somente com um turno, podem se beneficiar com este modelo. Outro exemplo, são sistemas conversacionais que atuam com perguntas e respostas, onde as respostas geralmente possuem termos que são similares a pergunta, e podem ser mais adequados para um modelo simples como o Espaço Vetorial.

Além disso, foi mostrado que a quantidade de dados necessárias para treinamento do modelo Dual Encoder LSTM, influencia em seu desempenho, principalmente quando treinado com poucos dados, tendo uma piora de aproximadamente 24% quando treinado com 10^3 de exemplos em relação a 10^6 de exemplos.

No que diz respeito às limitações identificadas durante o desenvolvimento deste trabalho, destaca-se a necessidade da utilização de processamento paralelo no treinamento do modelo de *Deep Learning*, pois antes de usar CUDA, o treinamento estava sendo lento e extenso. Além disso, como os 10 conjuntos de treinamento eram constituídos de 1.000.000 de exemplos, gera-los em um HD demandava muito tempo, pois conforme a quantidade de turnos aumentava, a pesquisa no diretório de diálogos era maior, fazendo com que está tarefa chegasse em casos extremos, a durar até mesmo, 36 horas. Dessa forma, fez-se necessário a utilização de um SSD que diminuiu o tempo de geração, para em média, 1 hora.

Esta pesquisa permitiu a aplicação de conhecimentos obtidos no decorrer de todo o curso de Bacharelado em Sistemas de Informação, principalmente em relação a assuntos das disciplinas de Introdução a Pesquisa em Sistemas de Informação, Linguagem de Programação, Estatística e Inteligência Artificial. Além disso, a pesquisa possibilitou o aprendizado de outras tecnologias, como a linguagem de programação Python e suas respectivas bibliotecas, o *framework* Tensorflow e a plataforma CUDA, além das sofisticadas arquiteturas de redes de *Deep Learning*, que proporcionou um aprofundamento de conhecimentos além daqueles obtidos durante a graduação.

4.2 RECOMENDAÇÕES

Ainda há espaço para diversas direções de pesquisas em trabalhos futuros, que foram notadas durante o desenvolvimento deste trabalho, dentre as quais destaca-se:

a) Necessidade de uma avaliação qualitativa das respostas fornecidas pelo modelo Dual Encoder LSTM, para determinar as principais causas de erro na tarefa de recuperação da melhor resposta para um contexto.

b) Variação do conjunto de teste, pois nesta pesquisa, um único conjunto de teste constituído de contextos de tamanhos variados foi utilizado. Por exemplo, se o modelo foi treinado com contextos de um turno, o conjunto de testes também deverá conter apenas contextos de um turno. Desse modo, avaliar o modelo Dual Encoder LSTM com exemplos de teste que possuam a mesma quantidade de turnos do conjunto de treinamento, pode ajudar a entender alguns aspectos de seu comportamento, como o desempenho obtido nesta pesquisa, que reduzia à medida que o contexto aumentava de tamanho.

c) Utilização de outras bases de diálogos a fim de obter um maior *benchmark* de comparação entre os modelos aqui discutidos e outras abordagens de aprendizado de máquina.

REFERÊNCIAS

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca**. 2.ed. Porto Alegre: Bookman Editora, 2013.

BALAKRISHNAN, V.; ETHEL, L.-Y. **Stemming and Lemmatization: A Comparison of Retrieval Performances**. Lecture Notes on Software Engineering, v. 2, n. 3, p. 262–267, 2014.

CAMBRIA, E.; WHITE, B. **Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]**. IEEE Computational Intelligence Magazine, v. 9, n. 2, p. 48–57, maio 2014.

CARDOSO, O. N. P. **Recuperação de Informação**. INFOCOMP, v. 2, n. 1, p. 33–38, 1 nov. 2004.

CHETLUR, S. et al. **cudnn: Efficient primitives for deep learning**. 2014. Disponível em: <http://arxiv.org/abs/1410.0759>. Acesso em: 5 jan. 2019.

CILIMKOVIC, M. **Neural networks and back propagation algorithm**. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, v. 15, 2015.

COLAH. **Understanding LSTM Networks**. 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 8 jan. 2019.

DENG, L.; YU, D. **Deep learning: methods and applications**. Foundations and Trends® in Signal Processing, v. 7, n. 3–4, p. 197–387, 2014.

FERNEDA, E. **Introdução aos modelos computacionais de recuperação de informação**. Rio de Janeiro: Ciência Moderna, 2012.

FISCHER, T.; KRAUSS, C. **Deep learning with long short-term memory networks for financial market predictions**. European Journal of Operational Research, v. 270, n. 2, p. 654–669, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [s.l.] MIT Press, 2016.

GURNEY, K. **An Introduction to Neural Networks**. London: CRC Press, 2014.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. Waltham: Elsevier, 2012.

HARTMANN, N. et al. **Portuguese word embeddings: Evaluating on word analogies and natural language tasks**. 2017. Disponível em: <http://arxiv.org/abs/1708.06025>. Acesso em: 5 jan. 2019.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. Porto Alegre: Bookman Editora, 2007.

HIJJAWI, M. et al. **ArabChat: an Arabic Conversational Agent**. Computer Science and Information Technology (CSIT), 2014 6th International Conference on. **Anais**...IEEE, 2014.

HIRSCHBERG, J.; MANNING, C. D. **Advances in natural language processing**. Science, v. 349, n. 6245, p. 261–266, 17 jul. 2015.

IKONOMAKIS, M.; KOTSIANTIS, S.; TAMPAKAS, V. **Text classification using machine learning techniques**. WSEAS transactions on computers, v. 4, n. 8, p. 966–974, 2005.

JI, Z.; LU, Z.; LI, H. **An Information Retrieval Approach to Short Text Conversation**. 2014. Disponível em: <http://arxiv.org/abs/1408.6988>. Acesso em: 12 out. 2018.

KADLEC, R.; SCHMID, M.; KLEINDIENST, J. **Improved deep learning baselines for ubuntu corpus dialogs**. 2015. Disponível em: <http://arxiv.org/abs/1510.03753>. Acesso em: 8 jan. 2018.

KORDE, V. **Text Classification and Classifiers: A Survey**. International Journal of Artificial Intelligence & Applications, v. 3, n. 2, p. 85–99, 31 mar. 2012.

LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learning**. Nature, v. 521, n. 7553, p. 436–444, maio 2015.

LI, W.; MAK, B. K. W. **Recurrent Neural Network Language Model Adaptation Derived Document Vector**. 2016. Disponível em: <https://arxiv.org/abs/1611.00196>. Acesso em: 8 jan. 2019.

LOWE, R. et al. **The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems**. 2015. Disponível em: <https://arxiv.org/abs/1506.08909>. Acesso em: 12 out. 2018.

LOWE, R. et al. **On the Evaluation of Dialogue Systems with Next Utterance Classification**. 2016. Disponível em: <http://arxiv.org/abs/1605.05414>. Acesso em: 8 jan. 2019.

LOWE, R. T. et al. **Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus**. Dialogue & Discourse, v. 8, n. 1, p. 31–65, 31 jan. 2017.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**. 2009.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine Learning: An Artificial Intelligence Approach**. [s.l.] Springer Science & Business Media, 2013.

NIELSEN, M. A. **Neural networks and deep learning**. [s.l.] Determination press USA, 2015. v. 25.

NISHIDA, T. et al. **Conversational informatics: a data-intensive approach with emphasis on nonverbal communication**. Japan: Springer, 2014.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial**. 3.ed. Rio de Janeiro: Elsevier, 2013.

NURVITADHI, E. et al. **Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC**. Field Programmable Logic and Applications (FPL), 2016 26th International Conference on. **Anais**...IEEE, 2016.

PENNINGTON, J.; SOCHER, R.; MANNING, C. **Glove: Global vectors for word representation**. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

PRECHELT, L. **Early Stopping — But When?** In: MONTAVON, G.; ORR, G. B.; MÜLLER, K.-R. (Eds.). **Neural Networks: Tricks of the Trade**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. v. 7700p. 53–67.

RAM, A. et al. **Conversational ai: The science behind the alexa prize**. 2018. Disponível em: <https://arxiv.org/abs/1801.03604>. Acesso em: 10 jan. 2019.

SAK, H.; SENIOR, A.; BEAUFAYS, F. **Long short-term memory recurrent neural network architectures for large scale acoustic modeling**. Fifteenth annual conference of the international speech communication association. 2014a.

SAK, H.; SENIOR, A.; BEAUFAYS, F. **Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition**. 2014b. Disponível em: <https://arxiv.org/abs/1402.1128>. Acesso em: 8 jan. 2019.

SAKAI, T. et al. **Topic set size design with the evaluation measures for short text conversation**. Asia information retrieval symposium. **Anais**...Springer, 2015.

SAYGIN, A. P.; CICEKLI, I.; AKMAN, V. **Turing Test: 50 Years Later**. Minds and Machines, v. 10, n. 4, p. 463–518, 1 nov. 2000.

SCHMIDHUBER, J. **Deep learning in neural networks: An overview**. Neural networks, v. 61, p. 85–117, 2015.

SHANG, L. et al. **Overview of the NTCIR-12 Short Text Conversation Task.** NTCIR. 2016.

SUTSKEVER, I.; MARTENS, J.; HINTON, G. E. **Generating text with recurrent neural networks**. Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011.

VEEN, Fjodor Van. **The Neural Network Zoo**. 14 set. 2016. Disponível em: <https://www.asimovinstitute.org/neural-network-zoo/>. Acesso em: 5 jan. 2019.

WANG, H. et al. **A Dataset for Research on Short-Text Conversations**. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. **Anais**...Seattle, Washington, USA: Association for Computational Linguistics, out. 2013.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier Brasil, 2014.

WU, S.-H. et al. **CYUT Short Text Conversation System for NTCIR-12 STC**. 2016.

YAN, R. **“Chitty-Chitty-Chat Bot”: Deep Learning for Conversational AI.** IJCAI. 2018.

ZAZO, R. et al. **Language identification in short utterances using long short-term memory (LSTM) recurrent neural networks**. PloS one, v. 11, n. 1, 2016.