GUILHERME HENRIQUE EMILIORELI GIAROLA

MÓDULO DE INTERPRETAÇÃO TEXTUAL PARA RECONHECIMENTO DE INTENÇÕES EM LINGUAGEM NATURAL

1^a edição

LAVRAS – MG

2021

GUILHERME HENRIQUE EMILIORELI GIAROLA

MÓDULO DE INTERPRETAÇÃO TEXTUAL PARA RECONHECIMENTO DE INTENÇÕES EM LINGUAGEM NATURAL

1ª edição

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Programa de Graduação em Ciência da Computação para a obtenção do título de Bacharel.

Prof. Erick Galani Maziero
Orientador

LAVRAS – MG

2021

Giarola, Guilherme Henrique Emilioreli

Módulo de Interpretação Textual para Reconhecimento de Intenções em Linguagem Natural / . 1^a ed. rev., atual. e ampl. – Lavras : UFLA, 2021.

70 p. : il.

Trabalho de Conclusão de Curso–Universidade Federal de Lavras, 2021.

Orientador: Prof. Erick Galani Maziero.

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

GUILHERME HENRIQUE EMILIORELI GIAROLA

MÓDULO DE INTERPRETAÇÃO TEXTUAL PARA RECONHECIMENTO DE INTENÇÕES EM LINGUAGEM NATURAL

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Programa de Graduação em Ciência da Computação para a obtenção do título de Bacharel.

APROVADA em 10 de Maio de 2021.

Prof. Luiz Henrique de Campos Merschmann	UFLA
Prof. Paula Christina Figueira Cardoso	UFLA

Prof. Erick Galani Maziero
Orientador

LAVRAS – MG
2021

Dedico este trabalho a todos os mestres e doutores os quais fizeram parte e são, em parte, responsáveis pela trajetória de aprendizado e conhecimento que culminou neste trabalho.

RESUMO

Dispositivos e sistemas inteligentes têm se popularizado, principalmente pela capacidade de interação através da linguagem natural, como o Português. Entender a intenção de um usuário a partir de sua fala não é uma tarefa trivial e envolve um conjunto de técnicas que permitem a uma máquina capturar a fala de um usuário, convertê-la em texto e extrair a intenção do usuário. No contexto de uma casa inteligente, diversas funcionalidades estão à disposição dos moradores e, a maneira mais natural e intuitiva de interação é a linguagem natural. Nesse trabalho propõe-se a exploração de técnicas do Processamento de Linguagem Natural (PLN) e Inteligência Artificial (IA) para a geração de um módulo de interpretação textual para mapear a frase gerada por um usuário (e identificar sua intenção) a uma funcionalidade de uma casa. Para tanto, objetiva-se o uso das técnicas já conhecidas de *machine learning Logistic Regression, Support Vector Machines, K-Nearest Neighbors, Decision Trees* e *Gradient Boosting* e a utilização, também, de duas técnicas para extração de atributos, (TF-IDF e Word2Vec) de forma a gerar o módulo de interpretação textual. De forma a treinar estes modelos citados, foi tomado por base um córpus de frases em linguagem natural, as quais foram anotadas juntamente às intenções esperadas. Os dados levantados não foram suficientes para a inferência de um resultado definitivo. Entretanto, os relatórios de classificação gerados e os testes de validação estatística são capazes de orientar e auxiliar futuros projetos.

Palavras-chave: Processamento de Linguagem Natural. Inteligência Artificial. Aprendizado de Máquina.

ABSTRACT

Smart devices and systems have become more popular, mainly due to the ability to interact through natural language, such as Portuguese. Understanding a user's intention by his speech isn't a trivial task and involves many techniques that allow a machine to capture a user's speech, convert it into text and extract the user's intention. In the context of a smart house, several features are available to it's residents and the most natural and intuitive way of interaction is by natural language. In this work, it's proposed to explore Natural Language Processing (NLP) and Artificial Intelligence (AI) techniques to generate a textual interpretation module that maps a user-generated phrase (and identify it's intention) to a smart house's functionality. For this reason, the main objective is to use already known techniques of machine learning Logistic Regression, Support Vector Machines, K-Nearest Neighbors, Decision Trees and Gradient Boosting and also the usage of two attribute extraction techniques, (TF-IDF and Word2Vec) in order to generate the textual interpretation module. In order to train these models, it was used a corpus of phrases in natural language, which were annotated along with the expected intentions. The data collected wasn't sufficient for inferring a definitive result. However, the generated classification reports and the statistic validation tests are capable of orientating and helping future projects.

Keywords: Natural Language Processing. Artificial Intelligence. Machine Learning.

LISTA DE FIGURAS

Figura 4.1 – Exemplos de Vetores gerados pelo TF-IDF e pelo Word2Vec, a partir da sentença exemplo "Tá tudo muito alto aqui." 45
Figura 5.1 – Matriz de Confusão - SVM - Todo o Conjunto de Frases . . . 53
Figura 5.2 – Matriz de Confusão - Árvores de Decisão - Todo o Conjunto de Frases 54

LISTA DE TABELAS

Tabela 4.1 – Córpus de Frases 44

Tabela 4.2 – Parâmetros disponíveis para a hiper-parametrização e treinamento dos modelos. 47

Tabela 4.3 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções implícitas, utilizando TF-IDF. 47

Tabela 4.4 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções implícitas, utilizando Word2Vec. 48

Tabela 4.5 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções explícitas, utilizando TF-IDF. 48

Tabela 4.6 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções explícitas, utilizando Word2Vec. 48

Tabela 4.7 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores de todo o conjunto de frases, utilizando TF-IDF. 49

Tabela 4.8 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores de todo o conjunto de frases, utilizando Word2Vec. 49

Tabela 5.1 – Relatório de classificação para o conjunto de frases com todos os dados de teste, utilizando TF-IDF. 55

Tabela 5.2 – Resultados do teste de validação estatística para o conjunto de frases com todos os dados de teste, utilizando TF-IDF. 55

Tabela 5.3 – Relatório de classificação para o conjunto de frases com todos os dados de teste, utilizando Word2Vec. 56

Tabela 5.4 – Resultados do teste de validação estatística para o conjunto de frases com todos os dados de teste, utilizando Word2Vec. . . 56

Tabela 5.5 – Relatório de classificação para o conjunto de frases com intenções explícitas dos dados de teste, utilizando TF-IDF. 57
Tabela 5.6 – Resultados do teste de validação estatística para o conjunto de frases com intenções explícitas dos dados de teste, utilizando TF-IDF. 57
Tabela 5.7 – Relatório de classificação para o conjunto de frases com intenções explícitas dos dados de teste, utilizando Word2Vec. . . . 58
Tabela 5.8 – Resultados do teste de validação estatística para o conjunto de frases com intenções explícitas dos dados de teste, utilizando Word2Vec. 58
Tabela 5.9 – Relatório de classificação para o conjunto de frases com intenções implícitas dos dados de teste, utilizando TF-IDF. 59
Tabela 5.10 – Resultados do teste de validação estatística para o conjunto de frases com intenções implícitas dos dados de teste, utilizando TF-IDF. 59
Tabela 5.11 – Relatório de classificação para o conjunto de frases com intenções implícitas dos dados de teste, utilizando Word2Vec. . . . 60
Tabela 5.12 – Resultados do teste de validação estatística para o conjunto de frases com intenções implícitas dos dados de teste, utilizando Word2Vec. 60

SUMÁRIO

1	**INTRODUÇÃO**	17
2	**FUNDAMENTAÇÃO TEÓRICA**	21
2.1	**Processamento da Linguagem Natural**	21
2.1.1	**Principais tarefas do Processamento de Linguagem Natural**	22
2.2	**Inteligência Artificial**	26
2.2.1	**Abordagens simbólicas (regras)**	26
2.2.2	**Aprendizado de Máquina (Machine Learning)**	27
2.3	**Assistentes virtuais**	35
3	**TRABALHOS RELACIONADOS**	39
3.1	**Assistente Virtual Inteligente para a Integração e Gerenciamento de Dispositivos IoT**	39
3.2	**Assistente virtual para facilitar o autocuidado de pessoas mais velhas com diabetes tipo 2**	40
3.3	**Tutor Inteligente no Apoio ao Ensino à Distância**	40
4	**METODOLOGIA**	43
4.1	**Conjunto de Dados**	43
4.2	**Pré-Processamento dos Dados**	44
4.3	**Treinamento**	46
4.4	**Estratégia de avaliação e Métricas**	49
5	**RESULTADOS**	53
5.1	**Todo o conjunto de dados**	55
5.2	**Intenções explícitas**	57
5.3	**Intenções implícitas**	59
6	**CONCLUSÃO**	63
6.0.1	**Contribuições Práticas**	63
6.0.2	**Contribuições Teóricas**	63
6.0.3	**Trabalhos Futuros**	64

REFERÊNCIAS 65
APENDICE A – Código explicado 69

1 INTRODUÇÃO

Uma casa inteligente tem como base a comunicação e troca de informações entre suas tecnologias e dispositivos, bem como a exposição de informações aos seus usuários, gerando *feedback* útil quanto ao gerenciamento do ambiente doméstico. De forma geral, a motivação principal (ou o propósito) na existência de uma casa inteligente é a melhoria na experiência de vida dos moradores, de alguma forma, seja por meio de um conjunto de funcionalidades simples ativadas por voz ou até mesmo a gerência automatizada da segurança e das provisões da casa (WILSON; HARGREAVES; HAUXWELL-BALDWIN, 2015).

A automação residencial sempre foi um tema bastante comum quando se trata de ficção científica. Diferentemente de dispositivos como notebooks ou smartphones, aparelhos específicos, como a "Alexa Echo", da Amazon, quando colocados em um ambiente com outros aparatos conectáveis, se tornam aparelhos compartilhados, utilizados por diversas pessoas e capazes de realizar uma gama útil de tarefas (GEENG; ROESNER, 2019). A redução nos custos e maior facilidade para obtenção de dispositivos para controle de acesso, bem como uma maior utilização/conhecimento da chamada "Internet das Coisas"(IoT) acabaram por tornar esta realidade um pouco mais palpável ao dia-a-dia dentro de um número cada vez maior de casas, com o passar dos anos (UR; JUNG; SCHECHTER, 2013). O número de pessoas que precisam de assistência diariamente tende a crescer com o passar dos tempos, devido aos diversos avanços tecnológicos, científicos e à melhoria na qualidade de vida. As pessoas tendem a viver mais e a se deparar, por consequência, com situações difíceis de ser contornadas devido à idade e problemas de saúde. Além disso, não se deve deixar de lado os casos de pessoas com inabilidades motoras, funcionais e/ou mentais as quais impedem uma possível independência no seu dia-a-dia. Estes fatores nem sempre deveriam ser contornados por métodos mais antiquados para sua solução, como a presença constante de enfermeiros/auxiliares ou algum membro familiar a tomar conta de quem necessita

de amparo mesmo para tarefas simples, visto que tira a independência de alguém muitas vezes resistente à ajuda ou que teria plenas condições de seguir uma vida normal, quando aliado à tecnologia, por exemplo.

A maioria dos times de design, os quais trabalham com tecnologias assistivas, segue uma metodologia de design baseada no usuário. Esta é baseada no fato de que grande parte das dificuldades encontradas pelos usuários-alvo ocorrem mais vezes por conta do design da tecnologia, em vez da real funcionalidade da mesma. Entretanto, algumas das tecnologias podem ser adaptadas (ou mesmo melhoradas) em vista de torná-las de mais fácil utilização por determinados grupos de usuários (ORPWOOD et al., 2005).

Para tanto, uma forma intuitiva de se adaptar residências automatizadas às necessidades dos usuários-alvo é por meio da utilização de linguagem natural para a ativação de funcionalidades/realização de tarefas. Como citado anteriormente, os aparelhos que centralizam todo o sistema de automação possuem softwares controlados por meio da voz (como o Amazon Echo, por meio da "Alexa", ou o Google Nest, por meio do Google Assistant). Assim, mesmo utilizadores menos acostumados à tecnologia e/ou mais resistentes a mudanças, podem se comunicar e interagir com estas tecnologias de forma mais natural.

Deve-se avaliar, contudo, quais as implicações em se tentar realizar tais mudanças. Somente a ativação por voz não é suficiente para que os sistemas compreendam toda e qualquer intenção de um usuário. Jargões, "gírias"e mesmo frases com conteúdo implícito nem sempre são capazes de ser processadas corretamente por estes sistemas e isto se torna um problema, ainda que o intuito seja uma comunicação mais fácil com o sistema.

Neste trabalho, portanto, tem-se como motivação a aplicação de técnicas para interpretação textual, de forma a contornar situações nas quais a linguagem não é compreendida corretamente pelo sistema. O Processamento de Linguagem Natural fornece algumas possíveis abordagens que podem ser exploradas. A abor-

dagem mais simples se utiliza de regras determinadas empiricamente, de forma a identificar, mapear e realizar uma tarefa em específico, baseando-se somente no conjunto de palavras de entrada. Contudo, pela própria limitação do conjunto de regras, nem sempre é trivial otimizar e aplicar esse tipo de técnica, que pode não ser muito robusta. Outra alternativa é a utilização de *Machine Learning*, subárea da Inteligência Artificial. Por meio desta, pode-se tentar inferir a necessidade do usuário por meio de contexto e geração de modelos específicos. Entretanto, algoritmos tradicionais necessitam de grandes volumes de dados. Assim, pode-se utilizar, também, abordagens baseadas em "Modelos de Língua", pré-treinados em grandes volumes de texto. Esses modelos de língua podem ser utilizados de forma a melhorar os resultados de um classificador, mesmo quando o conjunto de dados de treinamento é pequeno.

Tem-se como objetivo principal a utilização de modelos baseados em *Machine Learning* para identificar intenções em frases de usuários, mapeando-as para funcionalidades de uma casa inteligente. Na busca por esse objetivo, outros objetivos específicos são explorados: i) descobrir qual a melhor forma de extrair atributos do texto e, ii) definir o algoritmo de aprendizado de máquina que é mais promissor para a tarefa.

Assim, utilizando-se de dados levantados anteriormente, (córpus com frases em linguagem natural) inicia-se pela separação dos mesmos quanto ao pertencimento às classes de frases explícitas e implícitas (quando o usuário diz explícita ou implicitamente qual funcionalidade deseja ativar/desativar/modificar). Em seguida, serão aplicados métodos de pré-processamento destes dados, em vista de utilizar diversas ferramentas de *Machine Learning* para obtenção de resultados. Além disto, será também efetuado um processo chamado de "hiperparametrização", para a busca pelos melhores parâmetros para cada algoritmo de aprendizado de máquina. Por fim, os resultados serão analisados, inclusive pela aplicação de testes estatísticos para a escolha da melhor abordagem para a tarefa.

Na Seção 2, Fundamentação Teórica, serão abordados temas e conceitos relevantes, os quais virão a nortear a compreensão deste trabalho. Em seguida, na Seção 3, alguns trabalhos relacionados são citados, de forma a dar alguns exemplos similares ao presente trabalho. Na Seção 4, Metodologia, as atividades relevantes ao escopo do projeto são levantadas e é explicada a forma com a qual se desenvolveu a ferramenta. Já quanto à Seção 5, os resultados são explicitados em forma de tabelas, bem como serão realizados comentários relevantes ao pleno entendimento das mesmas. Por fim, a conclusão vem a levantar as contribuições teóricas e práticas realizadas pelo desenvolvimento e escrita deste projeto, assim como citar possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Processamento da Linguagem Natural

O Processamento da Linguagem Natural (ou PLN) é uma área multidisciplinar (que envolve diversas subáreas da Ciência da Computação e Linguística Computacional, por exemplo) a qual trata dos diversos aspectos da comunicação humana, desde palavras até discursos inteiros. Diferentemente de linguagens artificialmente criadas, como linguagens de programação ou notações matemáticas, pode-se verificar uma evolução constante e orgânica nas linguagens faladas, à medida que passam de geração para geração. Tal característica inerente às linguagens faladas torna complexa a tarefa de se compor regras exatas para caracterizá-las. Para tanto, diversas técnicas podem ser utilizadas de forma a interpretar e gerar linguagem natural por dispositivos de computação (BIRD; KLEIN; LOPER, 2009). Alguns objetivos comumente vistos em PLN podem ser listados, como a extração de informações de textos discursivos, sumarização automática, análise de sentimentos, identificação de entidades nomeadas e tradução de máquina (VIEIRA; LOPES, 2010). Pode-se dizer, portanto, que o PLN tem como intuito fazer o computador se comunicar em linguagem humana (ou pelo menos entender a linguagem natural). Segundo (GONZALEZ; LIMA, 2003), existem quatro etapas as quais são abordadas ao se pensar no processamento da linguagem, que são:

- morfológica: está ligado à construção e classificação de palavras em classes morfológicas (como substantivos e verbos);
- sintática: trata do relacionamento entre palavras e das funções estruturais de cada uma nas frases. Além disto, trata da formação de sentenças e das relações entre frases;
- semântica: este nível se caracteriza por uma abstração maior da relação entre palavras e significados e, assim como no nível sintático, das relações necessárias para a formação de sentenças com significado; e

- pragmática/discursiva: tem um enfoque maior nos diferentes contextos aos quais se pode aplicar frases e sentenças e o quanto os mesmos afetam seus significados, considerando as intenções do produtor do texto.

Todas estas etapas possuem características próprias e dificuldades específicas. Contudo, cada aplicação de PLN pode ter a necessidade de se preocupar mais com um subconjunto destas. Uma aplicação a qual visa extrair significado de um texto, como na ferramenta desenvolvida neste trabalho, não precisa ter um foco expressivo em fonética e fonologia (níveis anteriores à morfologia, não citados anteriormente, pois estão mais relacionados a tarefas de reconhecimento de fala). Entretanto, as partes semântica e pragmática precisam de uma maior atenção. Existe uma clara diferenciação entre as técnicas utilizadas na linguagem falada em comparação à linguagem escrita. Entretanto, além disso, é possível ver diferenças evidentes entre as técnicas utilizadas em tipos de aplicação distintos, ao se avaliar os textos para os quais se destinam cada aplicação. Sob uma abordagem mais prática, as aplicações que se utilizam do Processamento da Linguagem Natural acabam por se dedicar a níveis específicos de forma a tornar o tratamento da língua menos complexo.

2.1.1 Principais tarefas do Processamento de Linguagem Natural

2.1.1.1 Interpretação de Linguagem Natural

A Interpretação de Linguagem Natural é um subtópico do PLN o qual se utiliza de Inteligência Artificial e Aprendizado de Máquina para conseguir extrair informações compreensíveis de textos discursivos. Tal interpretação torna possível a interação direta entre humano e máquina, sem que se necessite da formalização sintática das linguagens de computadores. Além disso, é possibilitada também a comunicação por parte da máquina também em Linguagem Natural. Todavia, é facilmente perceptível que somente a programação estruturada (com conjuntos de *ifs* e *elses*, por exemplo) não seria capaz de tais feitos. Para tanto, aplicações baseadas

em Inteligência Artificial conseguem gerar modelos de interpretação mais robustos e capturar informações não explícitas no texto, como intenções e sentimentos.

Com a ascensão de novas tecnologias baseadas em *Deep Learning*, bem como com o surgimento dos chamados *"word embeddings"*, (representações vetoriais de palavras e sentenças do texto) as técnicas de PLN passaram a contar cada vez mais com o próprio texto como entrada para um modelo, sem a necessidade de uma engenharia de atributos (*feature engineering*). Assim, modelos baseados em Redes Neurais Recorrentes, ou RNNs, (as quais fazem uso destes *"word embeddings"*) passaram a obter o novo "estado da arte"quando se trata de modelos para interpretação da linguagem. Além disso, essas Redes Neurais também são treinadas utilizando-se de grandes quantidades de texto.

Deep Learning é um campo do *Machine Learning*, o qual se utiliza de algoritmos inspirados no funcionamento do cérebro, as chamadas "Redes Neurais Artificiais". São um conjunto de algoritmos, treinados por meio de quantidades grandes de dados, os quais tentam simular determinados tipos de comportamento humano. Além disso, a forma de aprendizado destes algoritmos é similar à presente na mente humana. Ao receber novas informações, o cérebro tenta comparar ao que já conhece, de forma a absorver novos conceitos. O mesmo acontece às redes neurais usadas nestes casos.

As Redes Neurais Recorrentes, por sua vez, são tipos de Redes Neurais Artificiais as quais têm a capacidade de levar em conta dados temporais. Isto é, utilizando-se de dados de entrada anteriores, como uma espécie de memória, estas são capazes de influenciar as entradas e saídas atuais da rede. Futuros eventos também poderiam ser decisivos quanto a determinar as saídas atuais de uma RNN. Entretanto, por motivos óbvios, as Redes Neurais Recorrentes não podem levar estes dados em consideração.

As primeiras instâncias a utilizarem "word embeddings"e a serem pré-treinadas em quantidades grandes de texto foram os modelos obtidos por meio de

algoritmos como "Word2Vec"(MIKOLOV et al., 2013) e "GloVe"(PENNINGTON; SOCHER; MANNING, 2014). O Word2Vec é um método não supervisionado para construção de uma representação matricial das palavras do texto. Esse algoritmo pode-se utilizar de dois métodos: o "Skip-gram"e o "Continuous Bag of Words (CBOW)". O "Skip-gram"é um método que tem como objetivo encontrar representações de palavras as quais sejam mais adequadas para a predição de outras palavras em um mesmo contexto, frase ou sentença (BÉRARD et al., 2016). Contudo, pode-se encontrar mais de uma palavra a qual seja dada como resultado pelo algoritmo, o que acaba por reduzir bastante sua eficiência. Ao contrário do "Skip-gram", para que se possa treinar um modelo utilizando o método de "Continuous Bag of Words", é dado o contexto de uma palavra e o modelo tenta prever qual a palavra a qual deve preencher melhor aquele trecho de texto. Ou seja, o "Skip-gram"tenta prever o contexto a partir da palavra e o "Continuous Bag of Words", o contrário.

O Word2Vec, portanto, é uma rede neural, a qual é treinada por meio da utilização de pares de palavras (representando entrada e saída, respectivamente) em vista de se conseguir os valores dos pesos de cada neurônio da rede. Basicamente, a palavra a se tentar prever, assim como a palavra de entrada são transformadas em vetores os quais representam tais palavras. É calculada a saída da rede por meio da multiplicação destas representações vetoriais (vetor de entrada e os pesos correspondentes à palavra, pós-treinamento). Após passar pela camada de saída da rede, o resultado é submetido a um classificador de regressão. Este, por sua vez, retornará como saída a probabilidade de que a palavra de entrada seja a mais adequada para a predição (MCCORMICK, 2016).

2.1.1.2 Geração de Linguagem Natural

A Geração de Linguagem Natural é o "processo de produção de frases em linguagem natural que possibilitem a comunicação com humanos". Em suma, essa

subárea tem como intuito a geração automática de narrativas, as quais explicam, descrevem ou sumarizam dados de entrada, de forma com que consigam ser compreendidos e entendidos como textos produzidos em linguagem natural por seres humanos.

Esta tarefa ocupa uma parte crítica em sistemas orientados a diálogos, Inteligência Artificial Conversacional, por exemplo. Os modelos nos quais se baseavam os primeiros trabalhos nesta área eram baseados em regras de sistemas especialistas. Estes últimos requeriam que fossem formuladas regras de maneira exaustiva, fatos estes que reduziam bastante a flexibilidade de aplicação destas ferramentas a diferentes contextos (VARSHNEY et al., 2020).

Um exemplo são os algoritmos baseados na utilização de cadeias de Markov, uma das primeiras abordagens para geração de textos dentro da área. Este modelo tinha como base considerar a relação entre palavras de uma sentença, de forma a prever qual seria a suposta próxima palavra a completar a frase, como nos aplicativos de sugestão de palavras em teclados de smartphones, por exemplo.

Atualmente, a Geração de Linguagem Natural busca evitar abordagens baseadas em regras ou em frases específicas para tornar as respostas mais naturais. Uma das técnicas que podem ser citadas é a utilização de modelos baseados em redes neurais (profundas) recorrentes, as RNN's (TRAN; NGUYEN, 2017). Comparados a abordagens mais tradicionais, estas redes conseguem aprender por meio de quantidades grandes de texto, mostrando-se como sendo uma solução aceitável. Contudo, estes modelos podem apresentar problemas quanto a capturar dependências que se encontram distantes no texto, ou mesmo gerar respostas curtas e genéricas.

Neste trabalho, não se utilizou técnicas de geração de linguagem natural, apenas focou-se na identificação das intenções da fala de um usuário para ativação das funcionalidades de uma casa inteligente. Para tornar um assistente virtual mais

natural, é desejável que o mesmo realize a geração de frases de maneira natural, como exposto nesta seção.

2.2 Inteligência Artificial

A Inteligência Artificial é um campo da Ciência da Computação, que tem como intuito desenvolver sistemas que simulem a capacidade humana na resolução de um problema. Utilizando-se de diferentes algoritmos e estratégias de decisão, aliados a grandes volumes de dados, sistemas de IA se mostram capazes de propor ações similares ao pensamento humano, quando solicitado. Um sistema de IA não só consegue manipular e/ou armazenar dados, como também é capaz de manipular, adquirir e representar conhecimento. Esta capacidade torna possível a realização de inferências, tendo como ponto de partida o que já é conhecido pela máquina, além de resolver problemas mais complexos por meio destas capacidades.

Hodiernamente, pode-se dizer que a IA está bastante presente nos mais diversos âmbitos do cotidiano, desde veículos autodirigidos até a automação de serviços financeiros. Um número considerável de sistemas e ferramentas já fazem uso da IA, seja qual for a finalidade. De forma a possibilitar esta "revolução"da IA, o armazenamento de dados cresceu consideravelmente, levando o conceito de "big data". Estes dados já influenciam e auxiliam áreas como serviços bancários e de atenção à saúde, por exemplo, (LOBO, 2018) por meio da organização e visualização de grandes conjuntos de dados de forma organizada, por exemplo, para que se possa descobrir padrões, avaliar de forma mais intuitiva estes dados e auxiliar na tomada de decisão.

2.2.1 Abordagens simbólicas (regras)

Popularizada com o surgimento dos Sistemas Especialistas e influenciada diretamente pela linguagem Prolog, é a abordagem mais tradicional da IA, introduzida por Newell e Simon, em 1976. Em sistemas baseados em IA, instrumentos

de computação processam símbolos em vez de letras e números. Esse tipo de sistemas processa cadeias de caracteres de forma a representar entidades ou conceitos do mundo real, sejam elas arranjadas em estruturas ou redes. Estas estruturas representam a relação entre símbolos e representações explícitas as quais possam agregar informação.

Quanto ao que se refere a aplicação, esta abordagem lida melhor com problemas bem definidos, nos quais a informação está toda disponível e presente, pouca ou nenhuma inferência deve ser feita e as regras a serem seguidas estão bem claras. Entretanto, ao tentar aplicar sistemas de I.A. baseados em regras para que possam tentar resolver problemas em áreas as quais requeiram conhecimentos implícitos, como por exemplo, a interpretação textual, tem-se uma grande limitação da abordagem baseada em regras,

Nesses casos, a quantidade de regras pré-estabelecidas aumenta de forma a inviabilizar a cobertura e tratamento de todos os aspectos destes problemas, fato este que quase torna impossível a utilização de abordagens simbólicas. Isso contrasta grandemente com o Aprendizado de Máquina, as quais tentam efetivamente aprender a partir de dados e inferir/extrair informação dos mesmos, possibilitando identificar padrões não tão evidentes nos dados.

2.2.2 Aprendizado de Máquina (Machine Learning)

O Aprendizado de Máquina é um subcampo da Inteligência Artificial que tem como objetivo desenvolver técnicas sobre aprendizado automático por meio da identificação de padrões em dados disponíveis sobre o problema a ser tratado, possibilitando a criação de sistemas capazes de aprender sem a intervenção humana. Um sistema de aprendizado pode ser descrito como um programa de computador, o qual "aprende"a tomar decisões baseado em experiências anteriores e em análises estatísticas.

Pode-se separar o Aprendizado de Máquina em três tipos: supervisionado, não-supervisionado e aprendizado por reforço. A aprendizagem supervisionada é a mais comum e intuitiva das três. Tenta-se prever uma variável, chamada "dependente"(classe) a partir de uma lista de outras variáveis "independentes"(atributos). Existem diversas técnicas já conhecidas e bem-sucedidas para a resolução de problemas utilizando de aprendizado supervisionado, como técnicas baseadas em regressão, redes neurais artificiais e árvores de decisão.

Entretanto, nem sempre é possível obter dados rotulados, ou seja, com as classes indicadas para cada instância dos dados de treinamento. E é nestes casos que se concentra o aprendizado não-supervisionado. Como não se tem dados rotulados, os algoritmos de aprendizado não-supervisionados buscam por grupos ou regras de associação, baseando-se nos atributos das instâncias de treinamento. Uma das limitações do aprendizado não-supervisionado é que os grupos (clusters) encontrados nem sempre podem corresponder diretamente às classes que se possa esperar para a resolução de um problema.

O terceiro e último tipo (por reforço) se baseia em aprender qual a melhor ação a se tomar, em vista das circunstâncias nas quais esta ação será executada. No aprendizado por reforço, um agente inteligente é obtido por meio das recompensas ou penalizações de suas ações, em busca de um objetivo. Portanto, nem sempre é tão imediata a aplicação de algoritmos de aprendizado por reforço em problemas comumente explorados no aprendizado supervisionado.

Algoritmos baseados em reforço necessitam de um ambiente, o conjunto de todas as ações que se pode efetuar dentro de um contexto, um estado, ou situação na qual o agente se encontra em um determinado momento e uma recompensa ao se atingir um estado específico. No caso, este agente realiza ações relacionadas ao ambiente e recebe uma recompensa, a qual pode ser maior ou menor, a depender da ação tomada, e tenta otimizar este valor para realizar a melhor ação em um determinado contexto.

O pré-processamento dos dados é uma etapa importante para se obter sucesso no aprendizado de máquina. Algumas técnicas utilizadas para a melhoria no desempenho e redução na dimensionalidade dos dados são as chamadas "técnicas para seleção de atributos"(features). Em bases de dados com muitas dimensões, a maneira mais simples de se reduzir a complexidade dos dados, além de possibilitar uma limpeza prévia nos ruídos presentes nos mesmos é a seleção das melhores dimensões (ou atributos) para a tarefa em questão. Para tanto, são escolhidas somente as dimensões as quais contém informações relevantes, em vista de se solucionar o problema (KHALID; KHALIL; NASREEN, 2014).

Outra etapa importante, antes de realizar o treinamento de modelos de aprendizado de máquina, é a extração de atributos. Em se tratando de textos, as instâncias de treinamento nem sempre têm o mesmo comprimento, seja em caracteres ou em palavras. Assim, para o uso de algoritmos tradicionais de aprendizado de máquina, deve-se extrair os atributos do texto de forma a igual a dimensão de atributos de cada instância para o treinamento. Nesse trabalho, utilizou-se duas técnicas: as baseadas na técnica de "Bag of Words"e as que se baseiam em *word embeddings*. O primeiro modelo visa gerar vetores numéricos , os quais contém as palavras contidas em um determinado conjunto de dados. A princípio, é feita a definição do vocabulário (conjunto de todas as palavras distintas presentes nos dados). Então, é obtida a frequência de aparição das palavras por frase e gerado um vetor de tamanho igual ao vocabulário, no qual o valor numérico de cada posição corresponde à tal frequência. Essa representação é esparsa, pois uma frase conterá apenas algumas palavras de todo o vocabulário.

Entretanto, pode-se verificar que somente a conversão do conjunto textual para vetores numéricos nem sempre traz resultados tão expressivos ou corretos. Além disso, algoritmos diferentes de vetorização podem gerar resultados drasticamente diferentes. Uma forma de melhorar tais resultados, bem como obter a

importância de uma determinada palavra quanto ao conjunto completo do texto é a utilização do chamado "TF-IDF".

Para tanto, tal abordagem é capaz de verificar a "frequência do termo"(*Term Frequency*) e a "frequência inversa do documento"(*Inverse Document Frequency*), ou o quão difícil é encontrar um termo dentro do conjunto de palavras. Caso uma palavra apareça muito dentro do texto, é atribuída a ela uma pontuação baixa, dada a falta de relevância para o contexto. Caso contrário, uma pontuação alta é atribuída, de forma a destacar palavras distintas. Um exemplo de palavra a qual aparece diversas vezes no córpus utilizado neste trabalho e tem baixa relevância pode ser "luz", geralmente relacionada às intenções que acendem ou apagam a lâmpada, como em "Ligar a Luz.". Por outro lado, palavras como "friiiiio", que aparece somente em uma das frases, ("Que friiiiio!!!") tem maior relevância para a classificação.

A segunda técnica (*word embeddings*) também gera vetores para cada frase de treinamento. Esses vetores têm um tamanho pré-definido que independe do tamanho do vocabulário e é, portanto, uma representação densa, pois todas as dimensões contém valores diferentes de zero. Os *word embeddings* são gerados com base na maior quantidade de textos que se tenha à disposição, com o intuito de gerar vetores representativos das palavras de um vocabulário. No caso deste trabalho, dado que o córpus de treinamento é pequeno (como será visto mais à frente neste documento), um *word embedding* já treinado em uma grande quantidade de textos foi utilizado.

2.2.2.1 SVM - Máquina de Vetores de Suporte

O SVM é um algoritmo para aprendizado supervisionado, o qual pode ser utilizado para resolver problemas, tanto de regressão, quanto de classificação, embora seja utilizado, em sua maioria, para o último caso. Basicamente, os dados são colocados em um suposto "espaço"N-dimensional, em que N é correspondente

ao número de parâmetros utilizados para caracterizar cada elemento do conjunto de dados.

Basicamente, o algoritmo tenta encontrar uma separação entre as classes, como uma reta que separa dois conjuntos disjuntos em um gráfico de duas dimensões. Contudo, quanto maior o número de parâmetros, mais difícil (ou mesmo impossível) fica a visualização gráfica da funcionalidade do SVM. Após encontrar a melhor forma de segregar as classes, por meio do chamado "hiper-plano", o algoritmo tenta maximizar as distâncias entre os pontos mais próximos de cada classe, em relação à divisão ("margem"), em vista de obter uma melhor separação. Além disso, possíveis dados que sejam pontos "fora da curva"não são considerados na separação (RAY, 2017).

Ademais, existem casos em que os dados não podem ser separados quando em uma dimensionalidade mais baixa. O SVM, entretanto, possui um "kernel"o qual é uma função para pegar entradas não-separáveis linearmente e adaptá-las a um espaço de maior dimensionalidade. É realizada uma transformação destes dados, de forma a obter um problema separável pelo algoritmo, para então processá-los de acordo com as saídas definidas.

2.2.2.2 Algoritmo de Classificação Naive Bayes

O algoritmo "Naive Bayes"é um classificador probabilístico amplamente utilizado. Usualmente aplicado em ferramentas e algoritmos de Processamento de Linguagem Natural, pode-se utilizá-lo quando os atributos os quais descrevem cada instância do conjunto de dados forem (supostamente) independentes. Isto é, não possuam correlação entre si. Este algoritmo trata sobre probabilidade condicional, ou seja, qual a probabilidade de um evento ocorrer, dado que existe outro evento.

Para tanto, de forma a classificar uma entrada, o Naive Bayes considera o número de parâmetros e de classificações, segundo a seguinte fórmula em que se

tem y como uma classe e cada um dos dados caracterizados por x_1 a x_n como os valores das features:

$$P(y|x_1...x_n) = \frac{P(y)P(x_1,...x_n|y)}{P(x_1,...x_n)} \tag{2.1}$$

O algoritmo é somente capaz de considerar os parâmetros independentes dois a dois. Portanto, a probabilidade de um x_i qualquer precisa somente ser calculada em relação a y, já que os outros parâmetros não interferem na probabilidade do x_i em questão:

$$P(x_i|y,x_1...x_n) = P(x|y_i) \tag{2.2}$$

Além disto, também segundo a independência de parâmetros, tem-se, portanto:

$$P(x_1,...x_n|y) = \prod_{i=1}^{n} P(x_i|y) \tag{2.3}$$

Como $P(x_1,...x_n)$ pode ser calculada e é constante, o termo à esquerda da equação é diretamente proporcional ao numerador à direita:

$$P(y|x_1,...x_n) \propto \prod_{i=1}^{n} P(x_i|y) \tag{2.4}$$

Assim, o classificador tenta, por fim, escolher um "valor-alvo"(y) o qual maximize a probabilidade associada aos parâmetros passados:

$$\hat{y} = \underset{y}{argmax} P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.5}$$

Este valor, representa, portanto, a probabilidade de uma determinada entrada pertencer a uma das classes possíveis.

2.2.2.3 Árvore de Decisão

O algoritmo Árvore de Decisão trata da identificação de padrões em dados por meio da geração de "perguntas"sobre os parâmetros associados aos exemplos

do conjunto de treinamento. Cada uma das questões pode ser vista como um dos "nós internos"da árvore, os quais apontam para todas as respostas possíveis àquelas questões. Cada um dos dados de entrada, portanto, passa por estes nós, de forma com que, ao chegar a uma das "folhas"da árvore, estejam classificados como pertencentes à classe referenciada por este nó. Em alguns casos, os nós representam as probabilidades de que o dado de entrada pertença a uma determinada classe.

A construção de uma árvore de decisão pode ser realizada utilizando dados de treinamento, além de ser possível controlar a quantidade de níveis presentes na mesma. Contudo, tal prática pode vir a causar o chamado *overfitting*, que é uma especialização da árvore somente nos dados de treinamento e perdendo sua capacidade de generalização para outros dados do mesmo problema. Então, para evitar o *overfitting*, são "podados"os nós da árvore, até que o algoritmo tenha uma capacidade aceitável de generalizar para outros dados que não sejam os de treinamento. A utilização de validação cruzada nos elementos de treinamento pode ser utilizada para validar a capacidade de generalização da árvore (KINGSFORD; SALZBERG, 2008).

2.2.2.4 Gradient Boosting

Alguns algoritmos de aprendizado de máquina podem ser categorizados como "ensembles", que consistem na geração de vários "classificadores fracos", como árvores mais rasas (poucos níveis de profundidade". Contudo, por mais que os resultados desses "classificadores fracos"possuam acurácia abaixo do esperado, existem técnicas para melhorar as saídas preditivas destes modelos, combinando tais classificadores. O Gradient Boosting, como o próprio nome explicita, é uma técnica "boosting", do grupo de classificadores "ensemble".

Nas técnicas de "boosting", diversos modelos são treinados, sequencialmente, utilizando, se possível, os modelos treinados anteriormente. Ao final, são combinados de maneira determinística e a saída (classe ou regressão) é obtida. O

Gradient Boosting cria um modelo de árvores de decisão, com uma aproximação simples, e obtém o "erro", durante o treinamento. Ou seja, a diferença entre o valor real e o encontrado pelo primeiro modelo. Então, o próximo modelo é treinado e sofre ajustes, baseados nesta diferença. Esse ciclo é repetido até que se atinja um critério de parada, como por exemplo, um erro mínimo. E então, são somados os ajustes de todos os modelos fracos, de forma a se obter um modelo final. A ideia por trás deste algoritmo em específico é correlacionar ao máximo os novos modelos com o gradiente negativo da função que gera o "erro". Por isso, "Gradient Boosting"(NATEKIN; KNOLL, 2013).

2.2.2.5 Algoritmo Random Forests

Assim como o Gradient Boosting, o algoritmo "Random Forests também pertence ao grupo de classificadores "ensemble", sendo este, como o próprio nome sugere, uma combinação de árvores de decisão. Um número grande (k) de vetores randômicos de mesma distribuição destes modelos é gerado e cada um deles dá origem a uma Árvore de Decisão, baseada em um conjunto randômico dos dados de treinamento. A natureza e dimensão desse vetor depende da sua utilização para a construção dos modelos. Além disso, tais estruturas geradas possuem baixa correlação, devido à dupla natureza randômica dos dados e vetores escolhidos (BREIMAN, 1999).

Segundo a chamada "Lei dos Grandes Números", a geração de um número arbitrariamente grande de vetores randômicos (e de Árvores de Decisão) tende a se aproximar cada vez mais de um valor esperado. No caso, ao se gerar diversas destas estruturas, o valor que mais aparecer como resultado se torna o "eleito"como saída deste algoritmo de classificação. O Random Forests obtém a classe esperada (ou regressão) à partir da votação da maioria das árvores de decisão gerada.

2.2.2.6 Regressão Logística

A Regressão Logística é um modelo que pode ser utilizado ao se tentar explicar uma variável de saída (resposta categórica) em função de outras variáveis (qualitativas ou quantitativas, chamadas variáveis explicativas). Existem alguns tipos de Regressão Logística, sendo eles determinados pelo número de variáveis explicativas. Para os casos em que somente existe uma variável explicativa, (quantitativa) deve-se utilizar a Regressão Logística Simples e, para os demais casos, utiliza-se a Regressão Logística Múltipla. (ROCHA, 2014)

Ao se utilizar este último modelo, deve-se introduzir as chamadas "variáveis dummies", ou indicadoras de categoria, as quais possibilitam a caracterização em mais de duas classes. Para tanto, cada uma destas variáveis recebe as probabilidades de cada entrada pertencer à classe correspondente à sua categoria. Tais valores são comparados com base em uma "função degrau", de forma a obter dados discretos (0 ou 1). Além disto, utilizando-se dos dados gerados por meio desta Regressão, pode-se obter não só a classe mais adequada à entrada, como também as estimativas das probabilidades desta pertencer à mesma.

2.3 Assistentes virtuais

Assistentes virtuais são interfaces as quais possibilitam que usuários possam interagir com dispositivos "inteligentes"por meio da utilização da fala, de forma a prover respostas para determinadas perguntas feitas pelos usuários. Dispositivos de busca tradicionais tendem a retornar milhares e até milhões de resultados relevantes por busca. Entretanto, estes assistentes são designados para retornar um resultado em específico e informá-lo ao usuário em linguagem natural falada (NOBLES et al., 2020).

Alguns aparelhos vêm se tornando cada vez mais comuns nos ambientes domiciliares. Exemplos destes podem ser citados, como o Echo, com a Alexa, assistente virtual da Amazon, ou o Google Nest, com o Google Assistant. Estes

interagem com diversos outros aparelhos específicos, como lâmpadas ou câmeras e podem ser acessados por meio de diversos aparelhos como os citados, ou até celulares e computadores. Além disso, as interações com o usuário são humanizadas, em vista de se alcançar uma melhor comunicação entre uma ferramenta e os mais diversos tipos de usuário por meio frases em linguagem natural.

Devido à crescente interação entre diferentes dispositivos, é possível, por exemplo, conectar a Alexa a uma Smart Fridge, para que sejam viabilizadas as compras no serviço "Amazon Fresh". Este serviço possibilita que sejam realizadas compras em mercados utilizando somente a voz. Além disso, caso se escolha uma música no serviço "Amazon Music", de streaming, a geladeira reproduz a canção escolhida enquanto se cozinha, por exemplo (JONCO, 2015).

Apesar de todo o conforto, pode-se pensar na facilidade gerada como uma maneira de tornar mais fáceis algumas tarefas complicadas para ser realizadas por idosos ou pessoas com problemas motores, por exemplo. O leque de possibilidades o qual é criado por meio da comunicação criada pode vir a evitar situações desgastantes e/ou difíceis. A capacidade de interação entre dispositivos inteligentes possibilita desde a utilização de luzes, ventiladores e televisões até interações mais elaboradas como a já citada "Amazon Fresh".

Entretanto, a curva de aprendizado e/ou sua utilização nem sempre são triviais para parte dos usuários, como idosos ou pessoas com desabilidades visuais. É notável a dificuldade destes usuários para se adaptar a tecnologias já comuns e enraizadas no cotidiano, como calculadoras. Logo, ainda que sejam úteis e viabilizem uma menor dependência em vista de se realizar tarefas, a curva mínima de aprendizado nem sempre é alcançada. Para tanto, uma interface de comunicação mais próxima do natural, com o uso de linguagem natural, é indispensável para atingir o máximo de usuários. Nesse trabalho, portanto, além de intenções explícitas, busca-se gerar um modelo de interpretação de intenções implícitas (a

ser detalhada mais à frente, nessa monografia) para aumentar a cobertura na comunicação entre uma casa inteligente e seus usuários.

3 TRABALHOS RELACIONADOS

De forma similar ao apresentado neste documento, existem trabalhos os quais seguem algumas abordagens similares e/ou tentam solucionar questões relacionadas ao desenvolvimento e melhoria de assistentes virtuais em geral.

3.1 Assistente Virtual Inteligente para a Integração e Gerenciamento de Dispositivos IoT

Em (RIVEROS; SECCO; FERNÁNDES, 2017), foi realizado o desenvolvimento de um assistente virtual ,que fosse capaz de conectar e simplificar a utilização de alguns dispositivos de "IoT"(Internet of Things) por meio de um bate-papo (chat). Este assistente virtual foi desenvolvido em cinco etapas, as quais foram a criação de um serviço de nuvem, de forma a gerenciar todos os dispositivos e a troca de informações, a criação da interface de comunicação do assistente com este serviço, a criação e desenvolvimento da rede neural (rede de multicamadas Perceptron) necessária para o treinamento e aprendizado do assistente, o desenvolvimento da aplicação e a integração do assistente virtual com plataformas já conhecidas de envio e recebimento de mensagens. A rede, em si, foi treinada de maneira iterativa, por meio da entrada de dados contínua, de maneira a aperfeiçoar a mesma a cada inserção dos dados.

Toda a criação do serviço se baseou em intenções e termos. O primeiro foi necessário para a classificação de mensagens/termos quanto a um conjunto fixo de intenções as quais o serviço deve reconhecer (como, por exemplo, a intenção de desativar um sensor, ao se passar como entrada a frase "Desative o comando de voz.", ou mesmo ativar ou desativar a iluminação de um ambiente por meio da entrada "Ligar todas as luzes.". Isso permite que se realize uma melhor análise da intenção por parte do algoritmo, além de permitir que se junte a intenção reconhecida aos termos obtidos pelo sistema e classificados como relevantes. Assim, é obtida uma informação a qual faz sentido para as regras de negócio definidas pelo

sistema e que possa ser aplicada à rede neural implementada. Após o desenvolvimento e treinamento da rede, a avaliação dos resultados quanto ao escopo entre a pergunta realizada pelo usuário e a resposta correta teve acurácia superior a 80%.

3.2 Assistente virtual para facilitar o autocuidado de pessoas mais velhas com diabetes tipo 2

Apresenta-se, em (BUINHAS, 2018), uma solução (não implementada) para o autocuidado de idosos. Para tanto, foi sugerida a implementação e inclusão de um avatar de representação antropomórfica, o qual se propõe a desempenhar a função de assistente virtual, bem como a utilização de inteligência artificial. Tem-se como objetivo preencher, também, uma lacuna presente nas tele-consultas referentes à diabetes, de forma a dispor uma forma de responder às necessidades diárias do paciente.

De forma a permitir que o aplicativo disponha de inteligência, entretanto, a solução pensada não tomou como base uma abordagem relacionada ao aprendizado de máquina em si, mas sim que se baseia em um sistema de regras pré-estabelecidas, estas as quais têm em vista dar a possibilidade de criação de grupos de perguntas e respostas a serem geradas e utilizadas pelo sistema, ao se basear na interação constante com o usuário.

3.3 Tutor Inteligente no Apoio ao Ensino à Distância

Na tese de (NEVES, 2019), foi desenvolvido um sistema de perguntas e respostas, de forma a melhorar o comportamento de um tutor virtual para ensino à distância. Este sistema, por sua vez, tem por motivo principal possibilitar a resolução do problema de contato pessoal entre alunos e professor, além de reduzir situações normalmente decorrentes do ensino remoto, como a falta de motivação ou a desistência de uma disciplina/curso, por exemplo.

Este tutor tem como utilidade atualizar os alunos acerca das novidades presentes na plataforma utilizada pela faculdade, bem como ajudar a indicar os prazos de tarefas a ser realizadas, motivar o aluno com frases motivadoras e/ou de chamada de atenção e incentivar a melhoria do aproveitamento do aluno nas disciplinas de uma universidade.

Para que se fosse possível implementar e testar tal sistema, diversos serviços de perguntas e respostas já disponíveis no mercado foram testados, como o "Watson", da empresa IBM. Entretanto, nenhuma das versões não-pagas dos mesmos foi capaz de resolver o problema citado. A abordagem utilizada foi a implementação de um algoritmo chamado de "Damerau-Levenshtein distance". Esta, por sua vez, trata-se de uma abordagem empírica, a qual calcula o número mínimo de alterações necessárias, de forma a alterar uma palavra para outra.

Uma base de dados de perguntas foi disponibilizada, de forma a realizar as comparações entre as perguntas inseridas no assistente virtual. Assim, foi possível avaliar a similaridade entre as perguntas introduzidas pelos utilizadores e as frases (já respondidas) presentes na base de dados. Também foi definido um limite mínimo para que o valor obtido pela abordagem fosse comparado a alguma das perguntas disponíveis.

Foram citados, nesta seção, três trabalhos os quais possuem suas similaridades com a ferramenta implementada. Entretanto, somente o primeiro faz uso de técnicas relacionadas ao Processamento de Linguagem Natural, por meio da classificação de mensagens e termos quanto a um conjunto de intenções. Além disso, este é o que mais apresenta similaridades quanto ao presente trabalho, dada a relação com dispositivos de IoT e à já citada utilização de técnicas do PLN. Os demais são relacionados a determinados aspectos deste trabalho.

O segundo trabalho possui semelhanças quanto ao que se refere à implementação de um assistente virtual orientado a facilitar a interação humano-computador por meio de técnicas conhecidas da Inteligência Artificial. Entretanto,

este se baseia em um sistema de perguntas e respostas. Por fim, o terceiro trabalho também é relacionado, devido ao fato de ser, também, uma ferramenta dotada de artifícios da inteligência artificial e do PLN para o reconhecimento de frases, pré-processamento de dados textuais e para a resolução de um problema específico, no caso, reconhecer intenções dos alunos nas frases de entrada e relacioná-las a um conjunto de dados pré-estabelecido.

4 METODOLOGIA

Nesta seção, são apresentadas as metodologias e ferramentas utilizadas para a realização dos experimentos e verificação dos objetivos apresentados na introdução. Serão avaliados o conjunto de dados, bem como explicado como foram levantados e obtidos os mesmos e apresentadas as metodologias para o tratamento e pré-processamento dos dados. Em seguida, a seção de treinamento apresentará como foram treinados os modelos de *Machine Learning* e otimizados seus parâmetros. Por fim, serão apresentadas as estratégias utilizadas para avaliação e as métricas selecionadas para verificação dos resultados.

4.1 Conjunto de Dados

O conjunto de dados utilizado foi criado por um grupo composto por sete orientados do orientador deste TCC, no ano de 2019. Estes foram instruídos a pensar e criar 27 frases cada um, fossem elas formais ou informais, de forma a ativar/desativar um conjunto de funcionalidades presentes em uma hipotética casa inteligente. Cada um dos anotadores escreveu as frases em um documento de extensão .xlsx, assim como explicitou a intenção esperada pela utilização da mesma e a ação esperada a ser ativada. Dentre estas, estavam as ações de acender e apagar uma lâmpada, ligar e desligar um ventilador ou uma televisão, aumentar ou diminuir o volume da televisão, bem como trocar entre um canal e outro. Foram levantadas 41 frases relacionadas a funcionalidades da lâmpada, 43 relacionadas ao ventilador e 103 frases relacionadas a funcionalidades da televisão (volume, canais e ativação da mesma).

Além disso, foi pedido que algumas das frases contivessem intenções claras (explícitas), como na frase "Iluminar o ambiente"e outras fossem mais subjetivas (como em "Minha vista ta doendo com essa claridade.", por exemplo). Ao fim, foram geradas 187 frases, além de terem sido anotadas, para cada frase, a intenção contida e a funcionalidade ativada pela mesma (como mostrado na Tabela 4.1).

Tabela 4.1 – Córpus de Frases

Tipo de Frases	Quantidade de Frases
Quantidade Total de Frases	187
Frases com Intenções Implícitas	123
Frases com Intenções Explícitas	64

Fonte: Do autor (2019)

4.2 Pré-Processamento dos Dados

Os dados foram compilados e dispostos em um arquivo de extensão .csv, o qual possui colunas correspondentes às frases, tipo (explícito/implícito), resposta a ser dada pelo sistema e ação a ser realizada. Para facilitar o uso dos algoritmos de aprendizado de máquina, os dados foram carregados em uma estrutura tabular (*dataframe*) utilizando a biblioteca Pandas, de linguagem de programação Python. Um pré-processamento dos dados textuais também foi aplicado, com o objetivo de remover a acentuação e letras maiúsculas nas frases.

A biblioteca utilizada para as etapas de pré-processamento dos dados e treinamento de modelos de aprendizado de máquina foi a "Scikit-Learn"[1]. Os rótulos de cada frase, ou seja, a funcionalidade a ser ativada, por ser um atributo categórico, foi transformado em números pela classe LabelEncoder [2], da biblioteca citada anteriormente. Dessa mesma biblioteca, uma outra classe que gera representações TF-IDF [3] foi utilizada para extrair os vetores esparsos das frases.

Com relação ao uso de *word embeddings*, o modelo "Glove"de 100 dimensões pré-treinado em uma grande quantidade de textos foi utilizado. Esse modelo [4] é disponibilizado pelo Núcleo Interinstitucional de Linguística Computacional (NILC). Como os vetores retornados pelo modelo são para as palavras individualmente, para obter uma representação da frase, a média dos vetores foi utilizada

[1] https://scikit-learn.org/
[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[4] http://www.nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc

como representação para os algoritmos de aprendizado de máquina. A partir de uma sentença simples, como "Tá tudo muito alto aqui.", geram-se vetores numéricos diferentes, (como mostrado na Figura 4.1) a depender da técnica escolhida. Também por meio do Scikit-Learn, foram divididos os dados em treinamento e testes, por meio do método *train-test-split*.

Figura 4.1 – Exemplos de Vetores gerados pelo TF-IDF e pelo Word2Vec, a partir da sentença exemplo "Tá tudo muito alto aqui."

```
TF-IDF:
  (0, 1)        0.4472135954999579
  (0, 0)        0.4472135954999579
  (0, 2)        0.4472135954999579
  (0, 3)        0.4472135954999579
  (0, 4)        0.4472135954999579
Word2Vec:
[array([-3.05735856e-01, -1.77687511e-01, -3.31434488e-01, -8.48858356e-01,
        -2.96531498e-01,  2.51416373e-03, -2.96251982e-01, -3.19136649e-01,
        -2.19754994e-01, -1.29450485e-01,  5.66140153e-02,  3.39437336e-01,
         4.83064950e-02, -5.40972531e-01,  9.80334878e-02,  1.57491505e-01,
         2.48661831e-01,  2.47282162e-01,  3.02413344e-01,  8.17823946e-01,
         5.98798990e-01,  5.45873344e-02,  3.70278507e-01, -4.15192366e-01,
        -5.40721655e-01, -6.60332978e-01,  9.96828079e-04, -5.88498473e-01,
         3.47100645e-01,  1.65343249e+00,  7.23456666e-02,  3.34975809e-01,
        -1.12200655e-01,  2.26949170e-01,  2.55855173e-01, -4.32572335e-01,
         8.94918367e-02, -2.05582842e-01, -5.73872864e-01,  4.53648001e-01,
         1.19539507e-01, -1.24653161e-01, -5.87590002e-02, -3.10654491e-01,
         8.44568312e-02,  2.25505993e-01,  2.46850148e-01, -1.61891669e-01,
         3.53598207e-01, -3.10813487e-01,  4.71486837e-01,  2.47742847e-01,
        -1.26712337e-01, -1.65999815e-01, -6.43244207e-01,  3.91349709e-03,
         1.76276669e-01,  2.64565319e-01, -6.26248345e-02,  6.90976605e-02,
         3.30132008e-01,  7.25116581e-03, -1.61137164e-01, -8.47693309e-02,
         1.36339828e-01,  3.80424351e-01, -1.91302836e-01, -1.24164671e-01,
        -3.16611715e-02, -1.33707330e-01, -6.25191629e-02,  1.35678515e-01,
        -3.38430971e-01,  2.77326673e-01,  1.93733418e+00, -6.03988282e-02,
        -3.20273012e-01, -4.67110090e-02,  9.66323316e-02, -1.21484317e-01,
         3.10391515e-01, -1.64968312e+00, -2.88071990e-01, -1.07916945e-03,
        -7.59570003e-02,  3.42326313e-01, -6.28150487e-03, -3.31995517e-01,
         6.59533367e-02, -1.21327840e-01, -1.34113327e-01,  1.07134140e+00,
        -3.09738308e-01, -6.59636796e-01, -2.17656687e-01,  4.60913032e-01,
         6.01156831e-01,  4.65423197e-01,  1.62937438e+00,  3.10008854e-01],
      dtype=float32)]
```

4.3 Treinamento

Para abordar esse problema de classificação multi-classe, foram selecionados alguns já conhecidos algoritmos de aprendizado de máquina, a saber Regressão Logística, Máquina de Vetores de Suporte, Árvores de Decisão, K-Nearest Neighbors e Gradient Boosting.

Para que tais algoritmos obtenham o melhor desempenho no conjunto de dados utilizado, deve-se realizar a hiper-parametrização dos mesmos. Essa etapa consiste na escolha dos melhores valores para os parâmetros dos algoritmos. Para tanto, outra classe do Scikit-Learn (GridSearchCV) [5] foi utilizada. Como entrada para a hiper-parametrização, alguns dicionários com os possíveis valores dos parâmetros foram criados e, assim, selecionados para treinamento dos modelos. Tais parâmetros estão presentes na Tabela 4.2.

As Tabelas 4.3 e 4.4 indicam os parâmetros selecionados pelo processo de hiper-parametrização, apenas considerando o conjunto de frases com intenções implícitas obtido a partir da extração de atributos utilizando TF-IDF e Word2Vec, respectivamente. Deve-se notar, também, que a partir de um mesmo dataset, os valores dos parâmetros foram diferentes para cada técnica de extração de atributos.

Já as Tabelas 4.5 e 4.6 apresentam os valores de parâmetros selecionados ao considerar o conjunto de frases com intenções explícitas. Nota-se, também, as diferenças quanto à seleção e utilização de parâmetros. Por fim, as Tabelas 4.7 e 4.8 apresentam os parâmetros selecionados para todo o conjunto de frases, utilizando TF-IDF e Word2Vec.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Tabela 4.2 – Parâmetros disponíveis para a hiper-parametrização e treinamento dos modelos.

Método	**Parâmetros Disponíveis**
LogisticRegression	penalty: [L1, L2], C: [0.1, 0.5, 1, 10, 50], fit_intercept: [True, False], solver: [newton-cg, lbfgs, liblinear, sag, saga]
SVC	C: [0.1, 0.5, 1, 10, 20], kernel: [linear, poly, rbf, sigmoid], shrinking: [True, False]
KNeighborsClassifier	criterion: [gini, entropy], splitter: [best, random], min_samples_split: [1, 2, 4, 6], max_features: [auto, sqrt, log2]
DecisionTreeClassifier	neighbors: [9, 11, 13, 15], weights:[uniform, distance], max_algorithm: [auto, ball_tree, kd_tree], leaf_size: [15, 30, 45]
GradientBoostingClassifier	learning_rate: [0.05, 0.1, 0.25, 0.5], n_estimators: [50, 100, 200], criterion: [friedman_mse, mse], min_samples_split: [1, 2, 4], max_depth: [2, 3, 5, 7], max_features: [auto, sqrt, log2]

Fonte: Do autor (2021).

Tabela 4.3 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções implícitas, utilizando TF-IDF.

Método	**Parâmetros Selecionados**
Logistic Regression	C: 50, fit_intercept: False, penalty: L2, solver: newton-cg
SVC	C: 10, kernel: sigmoid, shrinking: True
KNeighborsClassifier	algorithm: auto, leaf_size: 15, n_neighbors: 9, weights: distance
DecisionTreeClassifier	criterion: gini, max_features: log2, min_samples_split: 4, splitter: best
GradientBoostingClassifier	criterion: friedman_mse, learning_rate: 0.1, max_depth: 5, max_features: log2, min_samples_split: 4, n_estimators: 50

Fonte: Do autor (2021).

Tabela 4.4 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções implícitas, utilizando Word2Vec.

Método	**Parâmetros Selecionados**
Logistic Regression	C: 10, fit_intercept: True, penalty: L1, solver: saga
SVC	C: 10, kernel: linear, shrinking: True
KNeighborsClassifier	algorithm: auto, leaf_size: 15, n_neighbors: 13, weights: distance
DecisionTreeClassifier	criterion: gini, max_features: auto, min_samples_split: 4, splitter: best
GradientBoostingClassifier	criterion: friedman_mse, learning_rate: 0.25, max_depth: 2, max_features: log2, min_samples_split: 4, n_estimators: 100

Fonte: Do autor (2021).

Tabela 4.5 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções explícitas, utilizando TF-IDF.

Método	**Parâmetros Selecionados**
Logistic Regression	C: 50, fit_intercept: False, penalty: L2, solver: newton-cg
SVC	C: 10, kernel: sigmoid, shrinking: True
KNeighborsClassifier	algorithm: auto, leaf_size: 15, n_neighbors: 9, weights: distance
DecisionTreeClassifier	criterion: gini, max_features: auto, min_samples_split: 4, splitter: best
GradientBoostingClassifier	criterion: friedman_mse, learning_rate: 0.05, max_depth: 3, max_features: sqrt, min_samples_split: 4, n_estimators: 100

Fonte: Do autor (2021).

Tabela 4.6 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores do conjunto de frases com intenções explícitas, utilizando Word2Vec.

Método	**Parâmetros Selecionados**
Logistic Regression	C: 50, fit_intercept: False, penalty: L2, solver: liblinear
SVC	C: 10, kernel: rbf, shrinking: True
KNeighborsClassifier	algorithm: auto, leaf_size: 15, n_neighbors: 11, weights: distance
DecisionTreeClassifier	criterion: entropy, max_features: log2, min_samples_split: 2, splitter: best
GradientBoostingClassifier	criterion: friedman_mse, learning_rate: 0.05, max_depth: 5, max_features: sqrt, min_samples_split: 2, n_estimators: 50

Fonte: Do autor (2021).

Tabela 4.7 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores de todo o conjunto de frases, utilizando TF-IDF.

Método	Parâmetros Selecionados
Logistic Regression	C: 10, fit_intercept: False, penalty: L2, solver: newton-cg
SVC	C: 10, kernel: linear, shrinking: True
KNeighborsClassifier	algorithm: auto, leaf_size: 15, n_neighbors: 13, weights: distance
DecisionTreeClassifier	criterion: entropy, max_features: auto, min_samples_split: 2, splitter: random
GradientBoostingClassifier	criterion: friedman_mse, learning_rate: 0.05, max_depth: 2, max_features: sqrt, min_samples_split: 2, n_estimators: 200

Fonte: Do autor (2021).

Tabela 4.8 – Parâmetros selecionados pelo GridSearchCV para cada um dos estimadores de todo o conjunto de frases, utilizando Word2Vec.

Método	Parâmetros Selecionados
Logistic Regression	C: 10, fit_intercept: True, penalty: L2, solver: liblinear
SVC	C: 10, kernel: linear, shrinking: True
KNeighborsClassifier	algorithm: auto, leaf_size: 15, n_neighbors: 9, weights: distance
DecisionTreeClassifier	criterion: gini, max_features: log2, min_samples_split: 2, splitter: random
GradientBoostingClassifier	criterion: mse, learning_rate: 0.25, max_depth: 7, max_features: sqrt, min_samples_split: 2, n_estimators: 200

Fonte: Do autor (2021).

A estratégia de avaliação durante a hiper-parametrização foi a cross-validação de 5 pastas. A classe GridSearchCV retorna o melhor classificador, dados os conjuntos de parâmetros disponíveis. Este foi aplicado nos dados de teste para obtenção dos valores finais (a serem detalhados na próxima seção).

4.4 Estratégia de avaliação e Métricas

De forma a se medir a validade dos modelos gerados pós-treinamento, foi utilizada a estratégia de validação cruzada. Esta estratégia, por sua vez, tem como função avaliar a capacidade de generalização de um determinado modelo, com base em um conjunto de dados. Como resultados, foram geradas as matrizes de confusão e foram obtidos os valores de precisão, cobertura e medida-F. Cada

uma das matrizes de confusão possibilitou verificar os erros e acertos de cada classificador. No caso dos erros, pode-se ver a classe confundida na predição.

Para que fossem calculados tais valores, foram utilizadas métricas clássicas aplicadas a problemas de classificação, são elas:

$$Accuracy = \frac{TP + TN}{(TP + FP + FN + TN)} \tag{4.1}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{4.2}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{4.3}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{(Precision + Recall)} \tag{4.4}$$

Quanto aos valores de resultado, tem-se que a precisão avalia quantos dados foram classificados corretamente, em comparação com todos os dados da classe a serem avaliados. A cobertura é uma métrica que indica, dentre os dados considerados como "Verdadeiro Positivo", (classificados corretamente) quantos o modelo conseguiu classificar corretamente. Já a medida-F é, basicamente, a média harmônica entre precisão e cobertura.

De forma a afirmar qual o melhor classificador em cada cenário, foi necessária aplicação de testes estatísticos. Foi utilizado, então, o procedimento "5x2cv paired t test"[6]. Este procedimento consiste em dividir cinco vezes o dataset em conjunto de treinamento e teste. Em cada uma das cinco iterações, o desempenho de dois dos modelos é comparado e os conjuntos de teste e treinamento são gerados diferentemente. Assim, pode-se obter a variância das diferenças (s), a estatística *t* e o valor *P* destas execuções, de forma a verificar se existe uma diferença

[6] http://rasbt.github.io/mlxtend/user_guide/evaluate/paired_ttest_5x2cv/

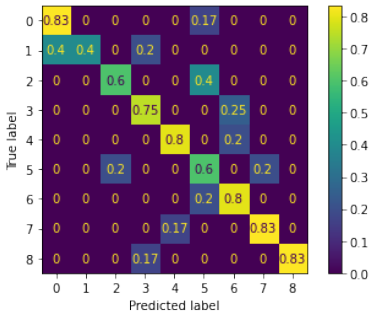
significante entre os dois modelos, baseando-se em um nível de significância de 95%.

5 RESULTADOS

De forma a verificar o desempenho de cada um dos algoritmos, são apresentadas as matrizes de confusão para cada um dos cenários, (frases implícitas, explícitas e todas) o que possibilita uma visualização mais simples e clara do comportamento dos classificadores treinados. Espera-se que a diagonal principal da matriz esteja completamente preenchida com o valor 1, ou seja, 100% dos valores preditos como pertencentes àquela determinada classe realmente pertencem à mesma. No caso de algum erro na classificação, os erros são exibidos fora da diagonal principal.

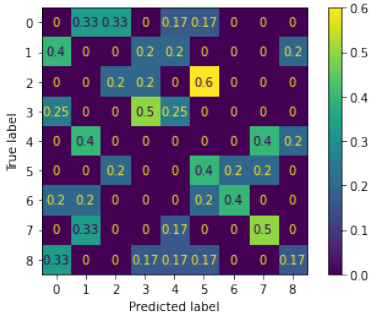
Entretanto, ao gerar essas matrizes, foram obtidos resultados muito diversos entre si. Alguns deles, como a matriz gerada pelo SVM para todo o conjunto de frases utilizando TF-IDF, foram os que melhor se aproximaram do resultado ótimo, como pode-se visualizar na Figura 1.

Figura 5.1 – Matriz de Confusão - SVM - Todo o Conjunto de Frases



Outros, entretanto, se mostraram menos relevantes, como a matriz gerada pelas Árvores de Decisão para todo o conjunto de frases utilizando Word2Vec como gerador de vetores (Figura 2).

Figura 5.2 – Matriz de Confusão - Árvores de Decisão - Todo o Conjunto de Frases



Após a hiper-parametrização, com base nos dados de teste, as métricas precisão, cobertura e medida-F foram obtidas. Por meio do método "classification_report"[7], foram obtidas as médias das métricas citadas. Estes resultados, por si só, não são capazes de afirmar se um modelo é significantemente melhor que outro. Para tanto, por fim foi necessária a aplicação de um teste pareado entre pares de modelos, de forma a obter os P-valores da comparação entre eles e determinar o melhor desempenho de um em detrimento dos outros.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

5.1 Todo o conjunto de dados

Tabela 5.1 – Relatório de classificação para o conjunto de frases com todos os dados de teste, utilizando TF-IDF.

Métricas	Logistic Regression	Support Vector Machines	K-Nearest Neighbors	Decision Trees	Gradient Boosting
Precisão (média)	75%	75%	55%	44%	65%
Cobertura (média)	72%	72%	55%	34%	58%
Medida-F (média)	71%	72%	54%	34%	59%
Acurácia	72%	72%	57%	34%	60%

Fonte: Do autor (2021).

Tabela 5.2 – Resultados do teste de validação estatística para o conjunto de frases com todos os dados de teste, utilizando TF-IDF.

Modelo 1 x Modelo 2	Estatística-T	P-Valor
Logistic Regression x SVM	-0.192	0.855
Logistic Regression x KNN	1.535	0.185
Logistic Regression x Decision Trees	3.294	0.022
Logistic Regression x Gradient Boosting	3.078	0.028
SVM x KNN	0.609	0.569
SVM x Decision Trees	2.953	0.032
SVM x Gradient Boosting	2.025	0.099
KNN x Decision Trees	4.055	0.010
KNN x Gradient Boosting	1.071	0.333
Decision Trees x Gradient Boosting	-2.651	0.045

Fonte: Do autor (2021).

Pode-se verificar, portanto, que todos os outros algoritmos são significantemente melhores que o algoritmo baseado em árvores de decisão, bem como o algoritmo Gradient Boosting é significantemente melhor que o algoritmo Decision Trees, mas tem desempenho pior que o Logistic Regression. Quanto às demais comparações, nada se pode afirmar, devido ao fato de que os P-valores não estão abaixo do limiar de significância (para $\alpha = 0.05$), segundo a Tabela 5.2.

Além disso, por meio do relatório de classificação, (Tabela 5.1) obtém-se que os modelos os quais obtiveram melhor desempenho foram o Logistic Regression e o Support Vector Machines (SVM), obtendo resultados de acima de 70% nas métricas avaliadas.

Tabela 5.3 – Relatório de classificação para o conjunto de frases com todos os dados de teste, utilizando Word2Vec.

Métricas	Logistic Regression	Support Vector Machines	K-Nearest Neighbors	Decision Trees	Gradient Boosting
Precisão (média)	67%	65%	36%	27%	40%
Cobertura (média)	65%	59%	29%	24%	41%
Medida-F (média)	64%	58%	28%	24%	38%
Acurácia	66%	57%	30%	23%	43%

Fonte: Do autor (2021).

Tabela 5.4 – Resultados do teste de validação estatística para o conjunto de frases com todos os dados de teste, utilizando Word2Vec.

Modelo 1 x Modelo 2	Estatística-T	P-Valor
Logistic Regression x SVM	-0.192	0.855
Logistic Regression x KNN	2.301	0.070
Logistic Regression x Decision Trees	6.689	0.001
Logistic Regression x Gradient Boosting	3.025	0.029
SVM x KNN	1.898	0.116
SVM x Decision Trees	3.630	0.015
SVM x Gradient Boosting	2.241	0.075
KNN x Decision Trees	2.553	0.051
KNN x Gradient Boosting	-0.338	0.749
Decision Trees x Gradient Boosting	-3.449	0.018

Fonte: Do autor (2021).

Ao se utilizar o Word2Vec para extrair os atributos do texto, entretanto, verifica-se que o Logistic Regression é significantemente melhor que o KNN, Decision Trees e o Gradient Boosting. Entretanto, não se pode garantir que o Logistic Regression seja melhor que o SVM, já que seu P-valor é de 0.855. O SVM, por sua vez, também é melhor que os outros três modelos, com significância de α = 0.05, segundo a Tabela 5.3.

Assim como os resultados obtidos baseando-se em TF-IDF, os modelos Logistic Regression e Support Vector Machines também obtiveram melhores resultados, quando comparados aos demais modelos. Entretanto, neste caso, o primeiro obteve melhores resultados nas quatro métricas (Tabela 5.4).

5.2 Intenções explícitas

Tabela 5.5 – Relatório de classificação para o conjunto de frases com intenções explícitas dos dados de teste, utilizando TF-IDF.

Métricas	Logistic Regression	Support Vector Machines	K-Nearest Neighbors	Decision Trees	Gradient Boosting
Precisão (média)	59%	57%	46%	36%	40%
Cobertura (média)	61%	57%	54%	46%	41%
Medida-F (média)	58%	56%	48%	37%	39%
Acurácia	60%	55%	55%	45%	40%

Fonte: Do autor (2021).

Tabela 5.6 – Resultados do teste de validação estatística para o conjunto de frases com intenções explícitas dos dados de teste, utilizando TF-IDF.

Modelo 1 x Modelo 2	Estatística-T	P-Valor
Logistic Regression x SVM	-0.650	0.544
Logistic Regression x KNN	2.887	0.034
Logistic Regression x Decision Trees	1.489	0.197
Logistic Regression x Gradient Boosting	1.505	0.193
SVM x KNN	2.053	0.095
SVM x Decision Trees	1.400	0.220
SVM x Gradient Boosting	1.394	0.222
KNN x Decision Trees	0.392	0.711
KNN x Gradient Boosting	-0.383	0.717
Decision Trees x Gradient Boosting	-0.578	0.588

Fonte: Do autor (2021).

Segundo a Tabela 5.6, apenas pode-se afirmar que os modelos Logistic Regression e SVM foram melhores que o KNN, já que seus P-Valores foram de 9.5% e 3.4%, abaixo do limiar estipulado para o valor de alpha ($\alpha = 0.05$).

A diferença nos valores quanto aos relatórios de classificação, entretanto, não foi tão expressiva quanto os relatórios apresentados para o conjunto de frases com intenções implícitas dos dados de teste. Os melhores resultados, neste caso, também foram Logistic Regression e Support Vector Machines. Porém, seus valores ficaram entre 55% e 60%, enquanto as outras métricas se mantiveram entre 36% e 55%, no melhor caso (segundo a Tabela 5.5).

Tabela 5.7 – Relatório de classificação para o conjunto de frases com intenções explícitas dos dados de teste, utilizando Word2Vec.

Métricas	Logistic Regression	Support Vector Machines	K-Nearest Neighbors	Decision Trees	Gradient Boosting
Precisão (média)	49%	64%	66%	40%	40%
Cobertura (média)	57%	59%	63%	37%	48%
Medida-F (média)	52%	60%	62%	35%	41%
Acurácia	60%	60%	65%	35%	50%

Fonte: Do autor (2021).

Tabela 5.8 – Resultados do teste de validação estatística para o conjunto de frases com intenções explícitas dos dados de teste, utilizando Word2Vec.

Modelo 1 x Modelo 2	Estatística-T	P-Valor
Logistic Regression x SVM	1.220	0.277
Logistic Regression x KNN	1.629	0.164
Logistic Regression x Decision Trees	4.727	0.005
Logistic Regression x Gradient Boosting	3.176	0.025
SVM x KNN	0.656	0.541
SVM x Decision Trees	5.120	0.004
SVM x Gradient Boosting	1.595	0.172
KNN x Decision Trees	3.171	0.025
KNN x Gradient Boosting	0.350	0.740
Decision Trees x Gradient Boosting	-3.674	0.014

Fonte: Do autor (2021).

Segundo a Tabela 5.8, pode-se verificar que o algoritmo Logistic Regression é significantemente melhor que o modelo Gradient Boosting. Além disso, quando comparado a todas as outras métricas, o Decision Trees performa de maneira inferior. Ao se avaliar os dados obtidos pelos relatórios de classificação, (Tabela 5.7) tem-se que tanto o SVM, quanto o KNN performaram melhor.

5.3 Intenções implícitas

Tabela 5.9 – Relatório de classificação para o conjunto de frases com intenções implícitas dos dados de teste, utilizando TF-IDF.

Métricas	Logistic Regression	Support Vector Machines	K-Nearest Neighbors	Decision Trees	Gradient Boosting
Precisão (média)	43%	49%	49%	26%	50%
Cobertura (média)	49%	49%	53%	27%	56%
Medida-F (média)	44%	44%	47%	25%	49%
Acurácia	52%	52%	56%	30%	59%

Fonte: Do autor (2021).

Tabela 5.10 – Resultados do teste de validação estatística para o conjunto de frases com intenções implícitas dos dados de teste, utilizando TF-IDF.

Modelo 1 x Modelo 2	Estatística-T	P-Valor
Logistic Regression x SVM	-0.343	0.746
Logistic Regression x KNN	0.791	0.465
Logistic Regression x Decision Trees	1.841	0.125
Logistic Regression x Gradient Boosting	0.000	1.000
SVM x KNN	1.348	0.235
SVM x Decision Trees	1.796	0.132
SVM x Gradient Boosting	0.303	0.774
KNN x Decision Trees	1.306	0.248
KNN x Gradient Boosting	-1.041	0.345
Decision Trees x Gradient Boosting	-2.449	0.058

Fonte: Do autor (2021).

Quanto às frases com intenções implícitas, ao se utilizar o TF-IDF, tem-se um conjunto de dados obtidos por meio dos relatórios de classificação, (Tabela 5.9) não sendo estes muito diferentes entre si. À primeira vista, pode-se observar que o algoritmo baseado em árvores de decisão retornou resultados abaixo das outras métricas.

Ademais, a tentativa de comprovação estatística da relevância de um ou mais métodos em comparação aos outros também foi muito pouco efetiva. O P-Valor que mais se aproximou da significância de $\alpha = 0.05$ foi o valor de 0.058, ao se comparar o modelo "Gradient Boosting"com o modelo baseado em árvores de decisão.

Tabela 5.11 – Relatório de classificação para o conjunto de frases com intenções implícitas dos dados de teste, utilizando Word2Vec.

Métricas	Logistic Regression	Support Vector Machines	K-Nearest Neighbors	Decision Trees	Gradient Boosting
Precisão (média)	38%	37%	43%	17%	36%
Cobertura (média)	44%	44%	42%	19%	40%
Medida-F (média)	39%	38%	39%	17%	36%
Acurácia	48%	48%	44%	19%	44%

Fonte: Do autor (2021).

Tabela 5.12 – Resultados do teste de validação estatística para o conjunto de frases com intenções implícitas dos dados de teste, utilizando Word2Vec.

Modelo 1 x Modelo 2	Estatística-T	P-Valor
Logistic Regression x SVM	1.026	0.352
Logistic Regression x KNN	3.038	0.029
Logistic Regression x Decision Trees	3.558	0.016
Logistic Regression x Gradient Boosting	2.928	0.033
SVM x KNN	1.065	0.335
SVM x Decision Trees	2.006	0.101
SVM x Gradient Boosting	3.162	0.025
KNN x Decision Trees	2.000	0.102
KNN x Gradient Boosting	0.000	1.000
Decision Trees x Gradient Boosting	-1.122	0.313

Fonte: Do autor (2021).

Por fim, os resultados do teste de validação estatística para o conjunto de frases com intenções implícitas (Tabela 5.12) pôde explicitar um melhor desempenho do modelo de regressão logística quando comparado aos modelos KNN, Decision Trees e Gradient Boosting. Além disso, o SVM também é significantemente melhor que o modelo Gradient Boosting, com um P-Valor de 0.025.

Todavia, ao se basear na Tabela 5.11, verifica-se que o único modelo o qual performou abaixo dos demais foi o baseado em árvores de decisão, com resultados próximos de 18%, enquanto os demais valores das métricas ficaram entre 36% e 48%.

Em suma, quanto a todo o conjunto de dados, pode-se verificar que os algoritmos *Logistic Regression* e *Support Vector Machines*, quando utilizados juntamente ao TF-IDF, obtiveram melhores resultados em seus relatórios de classi-

ficação. Ao se avaliar somente o conjunto de frases com intenções explícitas, entretanto, os modelos *Support Vector Machines* e *k-Nearest Neighbors* tiveram os melhores resultados quanto às métricas avaliadas, após o pré-processamento utilizando Word2Vec. Por fim, somente avaliando-se as intenções implícitas, *Gradient Boosting* e *K-Nearest Neighbors* tiveram melhor performance, utilizando TF-IDF.

Além disso, devem ser notadas as limitações da utilização de modelos baseados em Machine Learning e um córpus pequeno. Os resultados ao se treinar um modelo com um conjunto de 187 frases e tentar classificá-las em 9 diferentes classes são muito inconsistentes e vulneráveis a ruído. Ao se utilizar um conjunto de dados de teste de 25%, por exemplo, têm-se menos que 45 frases. Caso estas estejam, hipoteticamente, igualmente divididas entre o conjunto de classes, qualquer erro na classificação reduz em 20% o desempenho do modelo para aquela classe. Com um conjunto mais robusto de dados, pode-se melhorar o processo de treinamento e obter resultados mais precisos.

6 CONCLUSÃO

Este trabalho abordou a tarefa de identificar intenções, tanto explícitas quanto implícitas em frases de usuários de uma casa inteligente. Atrelando as áreas de Processamento da Linguagem Natural e Inteligência Artificial, tratou desde a criação de um córpus, pré-processamento, treinamento e validação de modelos de aprendizado de máquina.

Embora à primeira vista os resultados obtidos pelo modelo baseados em regressão logística ou mesmo o SVM tenham alcançado resultados expressivos, (em média, 72%) somente foi possível inferir que estes performam melhor que o algoritmo baseado em árvores de decisão (ao se utilizar o conjunto de dados completo e após a aplicação do TF-IDF). Não se mostrou possível, segundo os experimentos realizados, definir um ou outro modelo que performe definitivamente melhor que os outros quatro, bem como não foi possível descobrir a melhor forma de extrair atributos do texto.

6.0.1 Contribuições Práticas

De forma a melhorar ou mesmo cobrir lacunas em futuros projetos, pode-se utilizar o código desenvolvido durante este trabalho, em vista de se realizar a classificação de frases em português quanto a um conjunto limitado de classes bem definidas. Toda a ferramenta tem como base a leitura de um arquivo de extensão ".csv"para a obtenção dos dados. Logo, é possível, com poucas modificações, o código desenvolvido seja capaz de ser aplicado a sistemas e/ou servir para funcionalidades bem distintas do escopo de uma casa inteligente, por exemplo.

6.0.2 Contribuições Teóricas

O presente trabalho foi capaz de comparar cinco modelos diferentes de classificação amplamente utilizados pela área de aprendizado de máquina, de forma a verificar seu desempenho ao se tentar classificar e ativar um conjunto de possí-

veis funcionalidades em uma casa inteligente. Além disto, de forma a validar e comparar os dados, foi gerado um conjunto de tabelas de resultados, as quais são capazes de auxiliar possíveis novos experimentos na área, bem como servir de guia quanto à escolha entre modelos do Machine Learning para tarefas relacionadas à classificação de dados textuais.

6.0.3 Trabalhos Futuros

Outra implementação destas arquiteturas é o chamado "BERT"(Bidirectional Encoder Representation Transformers). A diferença principal deste modelo para os Transformers tradicionais é como ele olha a sequência de texto. Enquanto os outros modelos somente analisavam a sequência da esquerda para a direita ou combinavam treinamentos da esquerda para a direita ou o contrário, o BERT treina bidirecionalmente, de forma a conseguir um melhor e mais profundo entendimento do contexto presente em uma frase (DEVLIN et al., 2019).

Então, quanto a possíveis evoluções desta ferramenta, pode-se, pensar na evolução do modelo gerado e dos métodos utilizados, em vista de se aplicar uma solução baseada em textitDeep Learning. Para tanto, a utilização do BERT (Bidirectional Encoder Representation Transformers) bem como a avaliação e comparação das métricas e técnicas escolhidas devem ser avaliadas e comparadas, para que seja possível a comparação e verificação dos resultados obtidos.

REFERÊNCIAS

BÉRARD, A. et al. Multivec: a multilingual and multilevel representation learning toolkit for nlp. In: . [S.l.: s.n.], 2016.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python: analyzing text with the natural language toolkit**. [S.l.]: "O'Reilly Media, Inc.", 2009.

BREIMAN, L. Random forests. **UC Berkeley TR567**, 1999.

BUINHAS, S. d. S. **Assistente virtual para facilitar o autocuidado de pessoas mais velhas com diabetes tipo 2**. Tese (Doutorado), 2018.

DEVLIN, J. et al. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. 2019.

GEENG, C.; ROESNER, F. Who's in control? interactions in multi-user smart homes. In: **Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2019. (CHI '19), p. 1–13. ISBN 9781450359702. Disponível em: <https://doi.org/10.1145/3290605.3300498>.

GONZALEZ, M.; LIMA, V. L. S. Recuperação de informação e processamento da linguagem natural. In: **XXIII Congresso da Sociedade Brasileira de Computaçao**. [S.l.: s.n.], 2003. v. 3, p. 347–395.

JONCO, C. M. Hey siri: inteligência artificial e a humanização dos assistentes pessoais. Universidade do Vale do Rio dos Sinos, 2015.

KHALID, S.; KHALIL, T.; NASREEN, S. A survey of feature selection and feature extraction techniques in machine learning. In: IEEE. **2014 science and information conference**. [S.l.], 2014. p. 372–378.

KINGSFORD, C.; SALZBERG, S. L. What are decision trees? **Nature biotechnology**, Nature Publishing Group, v. 26, n. 9, p. 1011–1013, 2008.

LOBO, L. C. Inteligência artificial, o futuro da medicina e a educação médica. **Revista Brasileira de Educação Médica**, SciELO Brasil, v. 42, n. 3, p. 3–8, 2018.

MCCORMICK, C. **Word2vec tutorial-the skip-gram model**. [S.l.]: Retrieved, 2016.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. **arXiv preprint arXiv:1310.4546**, 2013.

NATEKIN, A.; KNOLL, A. Gradient boosting machines, a tutorial. **Frontiers in Neurorobotics**, v. 7, p. 21, 2013. ISSN 1662-5218. Disponível em: <https://www.frontiersin.org/article/10.3389/fnbot.2013.00021>.

NEVES, L. F. L. d. **Tutor inteligente no apoio ao ensino à distância**. Tese (Doutorado), 2019.

NOBLES, A. L. et al. Responses to addiction help-seeking from alexa, siri, google assistant, cortana, and bixby intelligent virtual assistants. **NPJ digital medicine**, Nature Publishing Group, v. 3, n. 1, p. 1–3, 2020.

ORPWOOD, R. et al. The design of smart homes for people with dementia—user-interface aspects. **Universal Access in the information society**, Springer, v. 4, n. 2, p. 156–164, 2005.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543.

RAY, S. **Understanding Support Vector Machine(SVM) algorithm from examples (along with code)**. 2017. <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>.

RIVEROS, L. J. M.; SECCO, A. L.; FERNÁNDES, C. M. R. Assistente virtual inteligente para a integração e gerenciamento de dispositivos iot. **Anuário Pesquisa e Extensão Unoesc Videira**, v. 2, p. e15225–e15225, 2017.

ROCHA, A. L. M. M. d. Regressão logística multinível: uma aplicação de modelos lineares generalizados mistos. 2014.

TRAN, V.-K.; NGUYEN, L.-M. Natural language generation for spoken dialogue system using rnn encoder-decoder networks. **arXiv preprint arXiv:1706.00139**, 2017.

UR, B.; JUNG, J.; SCHECHTER, S. The current state of access control for smart devices in homes. In: HUPS 2014. **Workshop on Home Usable Privacy and Security (HUPS)**. [S.l.], 2013. v. 29, p. 209–218.

VARSHNEY, D. et al. Natural language generation using transformer network in an open-domain setting. In: SPRINGER. **International Conference on Applications of Natural Language to Information Systems**. [S.l.], 2020. p. 82–93.

VIEIRA, R.; LOPES, L. Processamento de linguagem natural e o tratamento computacional de linguagens científicas. **EM CORPORA**, p. 183, 2010.

WILSON, C.; HARGREAVES, T.; HAUXWELL-BALDWIN, R. Smart homes and their users: a systematic analysis and key challenges. **Personal and Ubiquitous Computing**, Springer, v. 19, n. 2, p. 463–476, 2015.

APÊNDICE A – Código explicado

Com auxílio da biblioteca Pandas, o arquivo contendo as frases e suas anotações é carregado em um *dataframe* (arquivo de formato tabular). Para cada frase, a acentuação é removida.

São selecionadas quatro das colunas do *dataframe* e atribuídas a variáveis correspondentes às frases do córpus (X), à ação a ser realizada (Y1), à resposta dada pelo sistema hipotético (Y2) e ao tipo da frase, explícito ou implícito (Y3). De forma a facilitar a separação dos dados quanto ao tipo, estas variáveis similares a estas, porém filtradas pelo tipo também foram criadas.

É realizado um pré-processamento nos dados de Y1, de forma a transformar todos os dados textuais em numéricos, por meio da classe "LabelEncoder", do SKLearn. Em seguida, podem ser utilizados dois métodos para transformar os dados de X em vetores numéricos: Word2Vec e TF-IDF. O SKLearn dispõe de uma classe para a transformação de dados textuais em numéricos por meio do TF-IDF (sklearn.feature_extraction.text.TfIdfVectorizer). O resultado da utilização da função "vectorizer.fit_transform(X)"é atribuído a uma variável chamada de "X_tfidf".

Caso seja necessário utilizar o Word2Vec para tal transformação, entretanto, pode-se usar uma *word embedding* pré-treinada obtida por meio do repositório de Word Embeddings do NILC (Núcleo Interinstitucional de Linguística Computacional). Utilizando-a, é possível obter os vetores numéricos e gerar outro vetor de frases (X_w2v) convertidas para vetores numéricos.

Os vetores X_w2v (ou X_tfidf) e Y1_encoded são divididos em conjuntos de treinamento e teste (X_train, X_test, y_train e y_test). Para que os melhores parâmetros sejam selecionados, alguns dicionários de parâmetros são utilizados pela ferramenta "GridSearchCV"do SKLearn. Assim que cada conjunto de parâmetros é selecionado, os melhores estimadores são obtidos e, então, utilizam-se os dados de teste para a obtenção dos relatórios de classificação de cada um dos modelos.

Finalmente, para fins da validação estatística, é utilizada a classe "paired_ttest_5x2cv", que tem a função de comparar dois modelos e obter os valores da Estatística-T e o P-Valor e verificar se um modelo é significante melhor que outro, estatisticamente. Foram comparados todos os métodos e os valores, exibidos na tela para fins comparativos.

Para mais informações sobre o código implementado, pode-se acessar o seguinte link do repositório no GitHub: https://github.com/guilhermegiarola/TCC. Duas versões estão disponíveis, uma com otimização de parâmetros e outra com a utilização de modelos com os parâmetros padrão dos modelos de aprendizado de máquina.