**UNIVERSIDADE FEDERAL DO CEARÁ**

**CAMPUS QUIXADÁ**

**CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**GABRIEL DE SOUSA BOTELHO**

# UTILIZAÇÃO DE BASES DE CONHECIMENTO E *WORD EMBEDDINGS* PARA ENRIQUECIMENTO DOS DADOS DE TREINAMENTO DE CHATBOTS CONVERSACIONAIS

**QUIXADÁ**

**2022**

GABRIEL DE SOUSA BOTELHO

# UTILIZAÇÃO DE BASES DE CONHECIMENTO E *WORD EMBEDDINGS* PARA ENRIQUECIMENTO DOS DADOS DE TREINAMENTO DE CHATBOTS CONVERSACIONAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Profa. Ma. Lívia Almada Cruz

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

B761u Botelho, Gabriel de Sousa Botelho.
Utilização de bases de conhecimento e word embeddings para enriquecimento dos dados de treinamento de chatbots conversacionais / Gabriel de Sousa Botelho Botelho. – 2022.
44 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2022.
Orientação: Profa. Ma. Lívia Almada Cruz.

1. Chatbot. 2. Word embedding. 3. Sinônimos. I. Título.

CDD 004

---

GABRIEL DE SOUSA BOTELHO

# UTILIZAÇÃO DE BASES DE CONHECIMENTO E *WORD EMBEDDINGS* PARA ENRIQUECIMENTO DOS DADOS DE TREINAMENTO DE CHATBOTS CONVERSACIONAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: ___/___/___.

BANCA EXAMINADORA

---

Profa. Ma. Lívia Almada Cruz (Orientadora)
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Marcos Antonio de Oliveira
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

## AGRADECIMENTOS

Agradeço primeiramente a Deus por ter estado comigo em todos os momentos, por ter me dado saúde e por guiar meu caminho na realização dos meus sonhos.

Agradeço também a minha família, que sonhou este sonho comigo, que me deram força para seguir firme nessa jornada, que fizeram o possível e impossível para que chegasse até onde cheguei. Em especial os meus pais, pelo amor, cuidado e apoio passado nestes anos distantes de casa. Agradeço aos meus irmãos Nathália e Thyago por serem modelos de referência para mim. Agradeço também ao meu tio Lula, que sempre apoiou meus estudos e por ter me apresentado ao campus da UFC Quixadá. Ao avô Acácio por todo seu amor, carinho e apoio. A minha avó Valmira, que infelizmente não conseguiu estar presente neste momento, mas que sempre será meu anjo da guarda. Sem vocês nada disso seria possível.

Agradeço também aos meus amigos que fiz nesta jornada, em especial Mateus, Roberta, Kassiane, Wallesson, Wesley Pedro e Karine. Sou grato por todo o apoio de vocês durante os estudos, pelas conversas, conselhos, festas e momentos de lazer. Vocês tornaram esses anos mais leve e inesquecíveis.

Agradeço também aos meus amigos de infância Regina e Luan, por mesmo distantes esteram sempre presentes e me apoiando.

Agradeço a minha orientadora Lívia Almada, por todo o conhecimento transmitido, pela paciência, compreensão e por ter me acolhido tão bem durante esses meses de trabalho.

Agradeço a todos professores que tive a oportunidade de estudar. Saibam que vocês transformam vidas para sempre através da educação, serei eternamente grato a todos.

Agradeço também a todos os servidores da UFC Quixadá, pelo suporte nas horas que precisamos, pela organização e por serem sempre solícitos.

Por fim, agradeço a mim e todas as versões de mim que um dia já fui. Estes anos foram de muitas alegrias e tristezas que enfrentei com a cabeça erguida, mesmo quando não tinha forças para tal. Sou grato por me dar espaço para evoluir, por ser quem sou e por quem um dia serei.

"We gather stones never knowing what they'll mean. Some to throw, some to make a diamond ring."

(Taylor Swift)

## RESUMO

Os sistemas de diálogos ou chatbots são programas de computador que utilizam dados para compreender e responder as perguntas feitas pelos usuários. No entanto, nem sempre o chatbot possui este conjunto de dados prontos para serem usados em suas fases iniciais. Desta forma, a falta de dados iniciais se torna uma questão recorrente na criação de chatbots. Este trabalho tem como objetivo realizar o aumento dos dados de entrada do chatbot ao incorporar palavras similares e/ou sinônimos através do uso de *Word Embeddings* e base de conhecimente léxico *WordNet*. Este aumento de dados é alcançado através do desenvolvimento de uma API (do ingês, *Application Programming Interface*). Os passos seguidos na metodologia inicia com a definição das tecnologias, seguido pela modelagem e implementação do modelo gerador de dados, após é realizada é a implementação do módulo de compreensão da linguagem natural. O quarto passo é o estudo de caso e por fim, é realizada a avaliação dos resultados dos experimentos realizados.



**Palavras-chave:** Chatbot. *Word embedding*. Base de conhecimento.

## ABSTRACT

Dialog systems or chatbots are computer programs that use data to understand and answer questions asked by users. However, the chatbot does not always have this set of data ready to be used in its initial phases. In this way, the lack of initial data becomes a recurring issue in the creation of chatbots. This work aims to increase the chatbot input data by incorporating similar words and/or synonyms through the use of Word Embeddings and WordNet lexical knowledge base. This data augmentation is achieved through the development of an Application Programming Interface. The steps followed in the methodology begin with the definition of technologies, followed by the modeling and implementation of the data generator model, after which the implementation of the Natural Language Understanding is carried out. The fourth step is the case study and finally, the evaluation of the results of the experiments performed.



**Keywords:** Chatbot. Word embedding. Knowledge base.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE QUADROS

**SUMÁRIO**

# 1 INTRODUÇÃO

A popularidade dos sistemas de diálogo vem aumentando ao passar dos anos. Já é possível encontrá-los em diversos meios como carros, televisão e telefones (FORGUES *et al.*, 2014). Os sistemas de diálogo ou agentes conversacionais são sistemas computacionais que, através da fala ou escrita, se comunicam com os humanos (MALLIOS; BOURBAKIS, 2016). Atualmente, os sistemas de diálogo conseguem conduzir um diálogo entre humano e máquina graças aos avanços obtidos no campo de Inteligência Artificial (IA) (MALLIOS; BOURBAKIS, 2016).

Este trabalho foca nos sistemas de diálogo em texto de chatbots. A tecnologia de chatbot incialmente foi utilizada para entretenimento, e atualmente é uma tecnologia difundida em diversas aplicações. A comunicação em um chatbot é obtida através da conversação entre o usuário e o computador. Essa comunição ocorre quando o usuário realiza uma pergunta em linguagem natural e o chatbot responde a pergunta em resposta linguagem natural. Assim, o chatbot tem um funcionamento similar ao de um mecanismo de pesquisa, se diferenciando ao retornar apenas uma saída para cada entrada.

Em Gapanyuk *et al.* (2018) são descritas quatro funções que os chatbot conseguem performar com sucesso, sendo elas perguntas e respostas, responder a frases comuns, diálogos de script e, por fim, resposta de base de conhecimentos (*Knowledge base*). Uma implementação típica de chatbot com base de conhecimento como cérebro contém um conjunto de modelos codificados manualmente que correspondem as entradas dos usuários e, que em seguida, geram as respostas (HUSSAIN; ATHULA, 2018). No entanto, a base de conhecimento muitas vezes conta com uma quantidade limitada de dados.

As expressões em linguagem natural humana possuem muitas maneiras de serem formadas, variando de pessoa para pessoa. Os sinônimos possibilitam que uma expressão seja formada com diferentes palavras, mas que mesmo assim, o sentido seja preservado. A técnica de *Word Embedding* consegue relacionar as palavras que são similares dentro de um vetor. Alguns algoritmos foram propostos para o aprendizado de *Word Embeddings*, também conhecido como representação distribuída de palavras ou vetor de palavras (FORGUES *et al.*, 2014). Esses *embeddings* codificam palavras em vetores assim como, palavras com representação similar, em vetores próximos ao da palavra (FORGUES *et al.*, 2014). Embora existam muitos algoritmos para induzir *Words Embeddings*, os objetivos do treinamento de um *Word Embedding* frequentemente envolve maximizar a similaridade de vetores que ocorrem em contextos semelhantes e minimizar

a similaridade com outros vetores de palavras (FORGUES *et al.*, 2014).

No entanto, modelar um chatbot para combinar respostas precisas e de forma consistente em relação à intenção e ao contexto das entradas dos usuários é uma tarefa desafiadora (CHEN *et al.*, 2020). A limitada quantidade de dados na base de conhecimento, torna o chatbot com um baixo desempenho, já que ele não consegue entender todas as variações das entradas que o usuário envia. Em alguns casos, os chatbots não possuem uma base de conhecimento inicial, tornando assim o desenvolvimento mais complicado.

Quando o chatbot não tem uma base de dados inicial com informações suficiente para serem usados, é necessário aplicar técnicas para aprimoramento e/ou obtenção de mais dados. Trabalhos como os de Forgues *et al.* (2014) investigam a utilização de *Word Embeddings* na classificação de texto. Seu objetivo principal é avaliar técnicas para melhorar a performance de sistemas de diálogos com poucos dados iniciais, o qual, após a avaliação, é alcançado. Forgues *et al.* (2014) também propõe uma alternativa de utilização de *vector extrema*, para substituir a média usual dos vetores de uma frase.

Desta forma, os chatbots que utilizam uma base de conhecimento que não foi aprimorada, possuem baixo desempenho devido a incapacidade de compreensão e formulação das diversas sentenças que são formadas de várias formas.

## 1.1 Objetivos

O entrave que os chatbots encontram de limitações de dados, é um dos fatores pelo qual é importante aprimorar a qualidade e quantidade de dados que irá alimentar sua base de conhecimento. Objetivo geral deste trabalho é realizar o aumento dos dados de entrada do chatbot através da incorporação de palavras similares e/ou sinônimos. Desta forma, para resolver este problema, este trabalho propõe uma aplicação para dar suporte a geração de dados utilizando tecnologias como bancos de dados léxicos e *Word Embeddings* para obtenção de palavras similares. Portanto, espera-se contribuir com uma técnica que pode ser aplicada no desenvolvimento dos dados de entrada de um chatbot, resultando em uma melhor performance do mesmo.

### *1.1.1 Objetivos Específicos*

1. Desenvolver uma aplicação que dá suporte na geração de dados de entrada do chatbot.

2. Incorporar sinônimos na geração dos dados para aprendizado do chatbot.
3. Avaliar o desempenho do componente de compreensão da linguagem natural de um chatbot com diferentes estratégias para obtenção dos sinônimos.

## 1.2 Estrutura do Trabalho

O restante do trabalho segue a seguinte estrutura. No Capítulo 2 é apresentado a fundamentação teórica, contendo os principais conceitos abordados neste trabalho. No Capítulo 3 são introduzidos os trabalhos relacionados a este. No Capítulo 4 é exposta a metodologia usada neste trabalho. No Capítulo 5 são definidas as tecnologias utilizadas no desenvolvimento da proposta. No Capítulo 6 é apresentada a arquitetura do gerador de dados. No Capítulo 7 é apresentado o estudo de caso e os resultados obtidos. Por fim, no Capítulo 8 é apresentada a conclusão deste trabalho.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados alguns conceitos necessários para o entendimento e desenvolvimento do trabalho proposto.

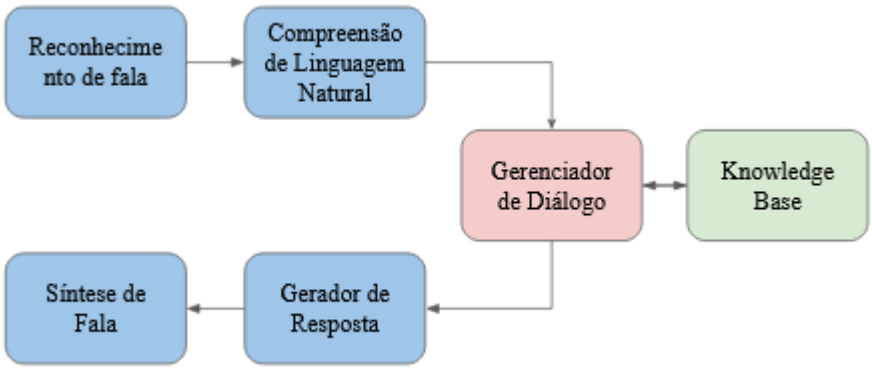
## 2.1 Chatbot

A comunicação entre humano e máquina se dá de diversas formas, sendo uma dessas, por meio de chatbots. O chatbot é um programa de *software* que é capaz de entender e responder as mensagens dos usuários, seja em forma de texto ou de voz. Sua intenção é simular o papel de um humano na conversação utilizando de linguagem natural para tornar a ação mais próxima da realidade.

Segundo Mason (2019), os chatbots podem ser definidos em algumas categorias: chatbot de Suporte, os quais atuam como ferramenta de suporte em um domínio específico, como os com conhecimentos sobre determinada companhia; existe também o chatbot de Habilidades, que define seus comandos de forma a facilitar a vida dos usuários, a funcionalidade de voz é indicada para esse chatbot; por fim, o chatbot Assistente, que é um meio termo entre os dos últimos, pois ele funciona melhor quando sabe um pouco sobre uma variedade de tópicos.

Os chatbots não seguem uma arquitetura padrão em seu desenvolvimento, eles podem variar de acordo com o que sua aplicação necessita. De acordo com Mallios e Bourbakis (2016) alguns dos componentes principais presentes nele são: Reconhecimento de fala, no caso de chatbots que recebem como entrada voz; Compreensão de Linguagem Natural, componente que converte a entrada de forma que o sistema entenda; Gerenciamento de Diálogo, o componente

Figura 1 – Principais componentes de um sistema de diálogo.




Fonte: Mallios e Bourbakis (2016)

fundamental em um sistema de diálogo já que ele analisa a entrada, mantém o histórico do diálogo, seleciona a estratégia de diálogo e controla o fluxo do diálogo; *Knowledge Base*, que armazena as informações que serão utilizadas pelo chatbot; Gerador de Respostas, retorna a saída para ser enviada ao usuário; Sintetizador de texto para voz, converte a saída gerada em forma de fala. Na Figura 1 é possível visualizar a ligação entre os componentes.

O módulo de compreensão de linguagem natural é o componente que este trabalho tem como foco e, para que seja possível avaliar este componente, é utilizado um *framework* de chatbot que é descrito no Capítulo 5.

## 2.2 Compreensão de Linguagem Natural

O processamento de linguagem natural (PLN) é um campo da Inteligência Artificial que dá às máquinas a habilidade de ler, entender e derivar significado de linguagens humanas (YSE, 2019). O PLN fornece ao chatbot a capacidade de realizar a comunicação entre humano-computador e computador-computador. É necessário proceder três análises para compreender a linguagem natural: Análise, essa etapa examina a estrutura sintáxica da frase e identifica suas principais relações linguísticas; Interpretação Semântica é a etapa encarregada de produzir a representação do significado dos texto operando o conhecimento do significado da palavra e da estrutura linguística; Estruturas Baseadas no Conhecimento (SETIAJI; WIBOWO, 2016), que armazenam o conhecimento do chatbot.

A Compreensão de Linguagem Natural *(NLU, do inglês Natural Language Understanding)* é um dos componentes presente na arquitetura do chatbot. O *NLU* é usado para extração relacional, paráfrase, inferência de linguagem natural, análise semântica, análise de sentimentos dentre outros (THAPA *et al.*, 2019). Segundo Mallios e Bourbakis (2016), o *NLU* converte as entradas dos usuários de forma elegível para o computador. Desta forma, o *NLU* fornece ao chatbot a capacidade de identificar qual a intenção e a entidade contida na mensagem sintática do usuário. A seguir são definidos os termos intenção e entidade.

**Intenção.** A intenção do usuário ou o propósito de sua mensagem é chamado de intenção (THAPA *et al.*, 2019). Em outras palavras, a intenção pode ser considerada um resumo do que o usuário deseja ou sua meta (THAPA *et al.*, 2019). Por exemplo, *"Quero o recheio de chocolate."*, a intenção pode ser *'pedir_recheio'*.

**Entidades.** As entidades são informações específicas e detalhadas que o usuário fornece (THAPA *et al.*, 2019). Por exemplo, na frase *"Quero o recheio de chocolate."*, *"chocolate"*

pode pertencer a entidade *'sabor_recheio'*.

De acordo com Abdellatif *et al.* (2021), para utilizar o *NLU* em um domínio específico é necessário que sejam definidos um conjunto de intenções e entidades personalizados para este domínio. Esses conjuntos devem ser treinados posteriormente para que o chatbot consiga reconhecê-los, caso contrário, a classificação errada da intenção e entidade pode impactar negativamente a experiência do usuário (ABDELLATIF *et al.*, 2021).

## 2.3 Representação Vetorial de Textos

De acordo com Mikolov *et al.* (2013), muitos sistemas e técnicas atuais de PLN tratam as palavras como unidades atômicas e não há noção de similaridade entre elas, já que são representadas como índices em um vocabulário. Essas técnicas possuem algumas vantagens como simplicidade e robustez, já que modelos simples treinados em grandes quantidades de dados tem uma performance superior em relação aos modelos complexos treinados em menos dados (MIKOLOV *et al.*, 2013). No entanto, segundo Mikolov *et al.* (2013), em algumas situações essas técnicas mais básicas não garantem um progresso significativo. Com o progresso das técnicas de aprendizado de máquina, tornou-se possível treinar modelos mais complexos em um conjunto de dados muito maior, e provavelmente o conceito mais bem sucedido é usar representações distribuídas de palavras, também conhecido vetores de palavras (MIKOLOV *et al.*, 2013).

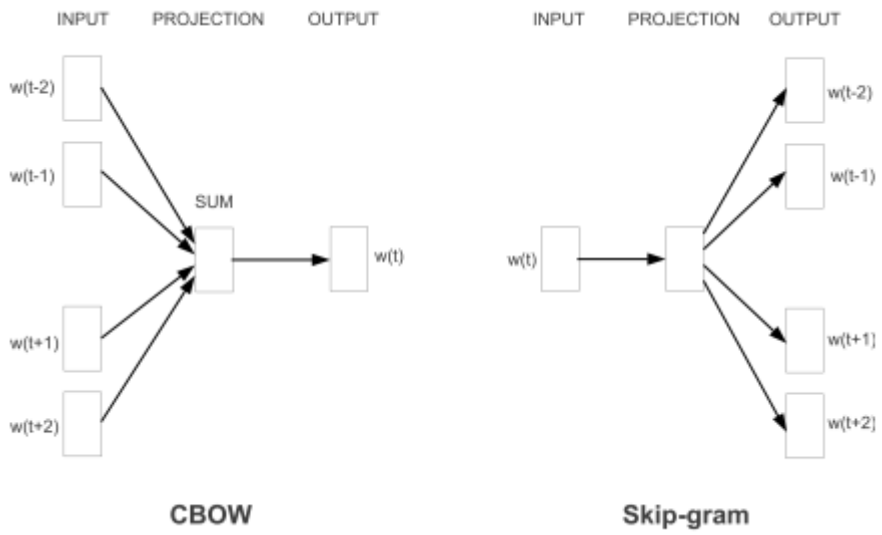
Os *Word Embeddings* são vetores de números reais, que representam palavras em um espaço *n*-dimensional, aprendidos a partir de grandes corpora não anotados e capazes de captar conhecimentos sintáticos, semânticos e morfológicos (HARTMANN *et al.*, 2017). Alguns algoritmos foram desenvolvidos para gerar *embeddings* e eles podem ser classificados em duas famílias de métodos: métodos que trabalham com uma matriz de ocorrência de palavras, como *Latent Semantic Analysis (LSA)*, *Hyperspace Analogue to Language (HAL)* e *Global Vectors (GloVe)*; O segundo é composto por métodos preditivos que tentam prever a palavra vizinha com base em uma ou mais palavras no contexto, como *Word2Vec* (HARTMANN *et al.*, 2017).

A seguir alguns modelos e técnicas que utilizam *Word Embeddings*.

### *2.3.1 Word2Vec*

O *Word2Vec* é um método amplamente usado em PLN na geração de *Word Embeddings* (HARTMANN *et al.*, 2017). Ele utiliza duas estratégias para o treinamento: *Continuous Bag-of-Words (CBOW)*, em que o modelo recebe uma sequência de palavras sem a do meio e tenta prever qual é essa palavra omitida; *Skip-Gram*, em que o modelo recebe uma palavra e tenta prever as palavras vizinhas (HARTMANN *et al.*, 2017). Cada estratégia é utilizada de forma separada de acordo com o problema ao qual é aplicada. Na Figura 2 é possível analisar a diferença entre a técnica utilizada em cada modelo.

Figura 2 – Imagem representativa da forma de predição de palavras dos modelos *CBOW* e *SkipGram*




Fonte: Mikolov *et al.* (2013)

### *2.3.2 FastText*

O método *FastText* associa os *embeddings* a *n-grams* de caracteres, de forma que a representação das palavras é dada pela a soma das representação da associação realizada (HARTMANN *et al.*, 2017). Assim, no *FastText* a concepção de palavra é induzida pela soma de vetores circundantes com vetores de caracteres *n-gram* (HARTMANN *et al.*, 2017). Desta forma, o *FastText* busca capturar informações morfológicas para induzir a incorporação de palavras (HARTMANN *et al.*, 2017).

### *2.3.3 Bidirectional Encoder Representations from Transformers*

*Bidirectional Encoder Representations from Transformers (BERT)* é projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente o lado esquerdo e direito do contexto (DEVLIN *et al.*, 2018). Existem duas etapas no desenvolvimento do *BERT*, sendo elas o pré-treinamento e ajuste fino. Segundo Devlin *et al.* (2018), no pré-treinamento o modelo é treinado em dados não rotulados em diferentes tarefas de pré-treinamento e no ajuste fino, o modelo *BERT* é inicializado primeiro com os parâmetros pré-treinados e todos os parâmetros são ajustados usando dados rotulados das tarefas *downstream*. Como resultado, o modelo *BERT* pré-treinado pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração para uma ampla variedade de tarefas de PNL (DEVLIN *et al.*, 2018).

## 2.4 Métricas de avaliação

As métricas de avaliação são importantes na compreensão dos resultados em um projeto. A partir delas, é possível afirmar se os resultados são satisfatórios ou não. Neste trabalho são utilizadas algumas métricas de avaliação para que seja possível avaliar os resultados da aplicação proposta. A seguir são definidas as métricas utilizadas neste trabalho.

**Acurácia**. A acurácia representa a razão entre o número de instâncias de dados classificadas corretamente e o número total de instâncias de dados.

**Precisão**. A precisão é a razão entre a quantidade de observações positivas previstas corretamente e o total de observações positivas previstas.

**Revocação**. A revocação é a razão entre as observações positivas corretamente previstas e todas as observações positivas que realmente são positivas.

***F1-Score***. A métrica *F1-Score* é a média harmônica entre a Precisão e Revocação.

# 3 TRABALHOS RELACIONADOS

A seguir são apresentado alguns trabalhos relacionados a este.

## 3.1 *Use of Word Embedding to generate similar words and misspellings for training purpose in chatbot development*

Em Thapa *et al.* (2019) é apresentada a proposta de geração de palavras similares e palavras com erros ortográficos, com o propósito de treinar o chatbot para entender os possíveis erros que ocorrem em um diálogo. Na geração de palavras com erros ortográficos é utilizada as técnicas de *word embedding* e *Levenshtein distance* ou distância editada. De acordo com Thapa *et al.* (2019), a distância editada é igual ao número mínimo de operações necessárias para transformar uma palavra na próxima palavra. Esta etapa é realizada através das operações da distância editada, que são a inserção, substituição e remoção de um caractere da palavra (THAPA *et al.*, 2019). Essas operações são aplicadas em um conjunto de 10 palavras pré definidas. Após a geração desses novos valores, é utilizado modelos de *word embedding* do *Twitter* com o intuito de capturar palavras que possuem erros ortográficos. Para isso, Thapa *et al.* (2019) recupera para cada uma das 10 palavras, 30 palavras similares e, ao final, seleciona somente as palavras que possuem o mesmo erro ortográfico das que foram geradas usando a distância editada. Ao final, somente as palavras iguais geradas pelas duas técnicas, são adicionadas aos dados de treino do chatbot.

A diferença entre Thapa *et al.* (2019) e este trabalho, está na geração de palavras similares através de modelos de *word embedding*, respeitando valores de similaridade mínima, e na geração utilizando de bases de conhecimento. Assim como, este trabalho tem como objetivo apresentar uma comparação do desempenho do modelo *NLU* sem obtenção de sinônimos e com a obtenção.

## 3.2 *Enriching Conversation Context in Retrieval-based Chatbots*

Em Vakili e Shakery (2019) foi proposta uma modificação para os *Bi-Encoders* do *BERT* na configuração de recuperação de respostas, a qual foi chamada enriquecimento de contexto. O enriquecimento de contexto compara não só apenas as respostas candidatas, como também, as representações do contexto da conversa. Desta forma, é comparado o contexto da conversa, com outros contextos mais semelhantes a conversa. Os experimentos envolvem

comparar a nova abordagem para o *BERT Bi-Encoder* e também dos métodos *Cross-Encoder* e *Bi-Encoder* que não usam *transformers* pré-treinados. Os modelos propostos foram implementados *PyTorch Framework* e para o *BERT* foi usado a implementação *Distilbert*, disponível na biblioteca *Huggingface's transformer*.

Algumas diferenças encontradas entre este trabalho e o Vakili e Shakery (2019), diz respeito a não utilização de semelhança de contexto, já que este trabalho foca em encontrar a palavras similares para geração de uma base de dados com mais conteúdo. Outro ponto, está na implementação dos modelos, o qual Vakili e Shakery (2019) utiliza o *PyTorch*, enquanto neste trabalho será utilizado o *framework Rasa*.

## 3.3 *Typographic-Based Data Augmentation to Improve a Question Retrieval in Short Dialogue System*

Em Khayrallah e Sedoc (2020) é proposto a utilização de sub-palavras para obter um melhor entendimento das palavras presentes nas perguntas, em comparação com os recursos que utilizam apenas as palavras completas. É realizado um incremento de dados baseado em sinônimos, abreviações e tipografia dos alfabetos do teclado *QWERTY* para lidar com as palavras fora de vocabulário. A estratégia utilizada consiste nos seguintes passos: Pré-processamento, que consiste em preparar o texto removendo pontuações, *stopwords*, colocar em *lower case* e *tokenização*; Aumento dos dados (*Data Augmentation*), que primeiro realiza a duplicação das perguntas trocando por palavras sinônimas e depois realiza o mesmo processo de duplicação mas, dessa vez, retira vogais e letras aleatórias de cada palavra para gerar abreviações e possíveis erros; Desenvolvimento do conjunto de treino e teste, nessa etapa é selecionado perguntas aleatórias das categorias para formar o conjunto e treino e teste; Aprendizado do *Word Embedding*, é usado os modelos, *Word2Vec* e *FastText*, no esquema que considera a informação das sub-palavras onde uma palavra é representada por uma bolsa de caracteres *n-grams*, dividindo as palavras em pedaços menores, possibilitando assim um bom estudo de palavras raras; Classificação da pergunta, que após de obter as representações das palavras desses vetores é calculada a similaridade de cosseno entre a pergunta armazenada e consulta, para que após sejam classificadas da maior até a menor similaridade; Métricas de avaliação, é usado o *mean reciprocal rank (MRR)* e o *Precision@N (P@N)*.

Assim como o Nugraha e Suyanto (2019), este trabalho irá usar modelos de *word embedding* na geração de palavras sinônimas, entretanto este trabalho utiliza apenas o modelo

*FastText* na recuperação. Ainda na recuperação de sinônimos, este trabalho utiliza da base de conhecimento léxico *WordNet* para obter sinônimos. Assim como, obtenção de sinônimos neste trabalho não foca em palavras com erros ortográficos, como forma de preservar a qualidade dos dados. Seguindo, este trabalho se diferencia no quesito de utilização de um *framework* no teste do conjunto de dados gerado. Em Nugraha e Suyanto (2019) não é afirmado se a proposta é testada em algum *framework* específico de chatbot, apenas é apresentado dados comparativos de desempenho de modelos *Skip-Gram* e *CBOW* no aprendizado, usando os dados originais e argumentados. Desta forma, neste trabalho o treino e teste do conjunto de dados gerados será realizado no *framework Rasa*, garantindo um melhor entendimento do comportamento do chatbot com os dados obtidos.

## 3.4 *SMRT Chatbots: Improving Non-Task-Oriented Dialog with Simulated Multiple Reference Training*

Em Khayrallah e Sedoc (2020) é proposto o modelo *Simulated Multiple Reference Training (SMRT)*, que usa de paráfrase para aproximar o espaço total de traduções possíveis das sentenças. Este processo é realizado da seguinte forma: treinando o modelo de tradução de máquina para prever a distribuição sobre os *tokens* possíveis da paráfrase e tomando a amostragem do *token* avaliado anteriormente na distribuição da paráfrase. O *SMRT* é capaz de capturar a diversidade sintática mas não consegue representar todas as variações semânticas. Em Khayrallah e Sedoc (2020) ao aplicar o *SMRT* em chatbots teve como resultado que foi obtida uma maior variedade lexical, assim como possui um desempenho tão bom quanto o pré-treinado em avaliação humana, no que diz respeito as medidas automáticas de diversidade e qualidade.

O trabalho de Khayrallah e Sedoc (2020), assim como este, busca aprimorar a qualidade das bases de conhecimento através da variação das perguntas e respostas, sem perder o sentido original. Para isso, este trabalho utiliza da ferramenta *chatette*, que gera uma variação de novas sentenças de acordo com os valores de entidades identificados nela. Outra diferença é que este trabalho utiliza de modelos de *word embedding* e da base de conhecimento léxico *WordNet* na geração das palavras sinônimas.

## 3.5 *A Natural Language Understanding Model COVID-19 based for chatbots*

Em Santos Júnior *et al.* (2021) é proposto uma avaliação de diferentes estratégias de *sentences embeddings* e *word embeddings* para o problema de descoberta de intenções em diálogos sobre *COVID-19*. Os experimentos são realizados com os *embeddings Bert LaBSE, FLAIR, BERTimbau, Glove* e *MUSE*. Outra contribuição proposta em Santos Júnior *et al.* (2021) é a criação de um modelo NLU que pode classificar as mensagens enviadas pelos usuários via chatbot, atribuindo intenções com as entidades definidas nas sentenças. Para alcançar estes objetivos, Santos Júnior *et al.* (2021) utiliza os diálogos de profissionais da saúde provindos da Plataforma de Serviço do Coronavírus do estado do Ceará, Brasil composto por 1.237 diálogos coletados de 1º de maio de 2020 à 6 de maio de 2020, com um total de 53.633 frases. A partir desses dados, é realizado os passos de limpeza de texto, que consiste em remover sentenças duplicadas dos diálogos, selecionar os diálogos que obtiveram uma avaliação dos pacientes igual a 10 e a remoção de sentenças que contém informações pessoais, número de telefone, *URL's*, *emoticons* e lugares. O segundo passo é realizar a extração de entidades utilizando a ferramenta *SINTOMATIC*, que identifica sintomas presentes em diálogos de texto em linguagem natural. O terceiro passo seguido é a extração das intenções relacionadas ao diagnóstico do paciente. Este passo é alcançado através da aplicação de métodos não supervisionados baseado em aprendizado compreendendo em, gerar vetores de *embedding*, clusterização dos vetores de *embedding* e rotulação de intenções. Este passo teve como resultado a identificação de cinco clusters principais que foram rotulados como *inform_medicine, request_inform, inform_symptoms, greeting, inform_diagnostic*, sendo assim as intenções identificadas nos dados. O quarto passo é o treino e avaliação do modelo *NLU* que, após extrair as entidades e intenções e anotá-las no dados, utiliza do componente *Rasa NLU* para treinar o modelo *NLU* gerado. Por fim, é implantado o modelo *NLU* em um modelo de conversação. Os resultados alcançados em Santos Júnior *et al.* (2021), mostram que o modelo de *embedding LaBSE* conseguiu classificar as cinco intenções identificadas, assim como alcançou uma acurácia de 0.869 quando treinado o modelo *NLU* no *Rasa*.

Em Santos Júnior *et al.* (2021) o objetivo está na identificação e avaliação das intenções e entidades nos diálogos analisados. Já este trabalho, foca na criação de uma aplicação que recebe estes dados anotados afim de automatizar a criação dos dados que vão fazer parte do modelo *NLU*. Assim como, este trabalho se diferencia na geração de sinônimos para entidades afim de aumentar a quantidade de dados no modelo *NLU*.

A seguir no Quadro 1 se encontra algumas características de cada trabalho relacionado e deste trabalho. Nela é comparada os principais pontos deste trabalho, como utilização de *word embeddings* na busca de palavras similares, utilização de bases de conhecimento na busca de palavras similares, *framework* de chatbot utilizado e automatização da criação do modelo *NLU*.

Quadro 1 – Quadro de comparação entre os trabalhos relacionados e este trabalho

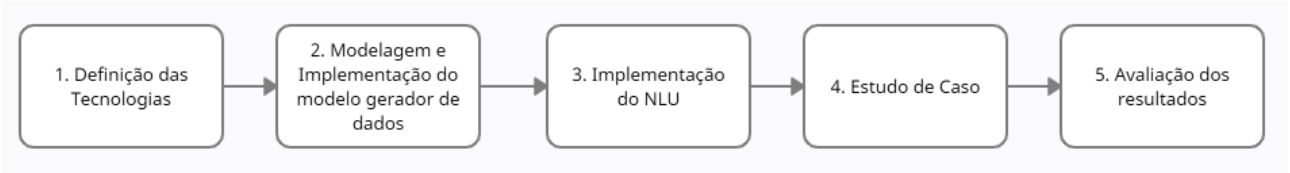
| Trabalho / Pontos de comparação | Usa *word embeddings* na busca de palavras similares | Usa bases de conhecimento na busca de palavras similares | *Framework* de chatbot | Métricas de avaliação do modelo *NLU* |
|---|---|---|---|---|
| Thapa *et al.* (2019) | *Word2Vec* | Não Utiliza | *Rasa* | Não especifica |
| Vakili e Shakery (2019) | *BERT* | Não Utiliza | *PyTorch* | Não especifica |
| Nugraha e Suyanto (2019) | *Word2Vec* e *FastText* | Não Utiliza | Não especifica | *Mean Reciprocal Rank (MRR)* e *Precision@N (P@N)* |
| Khayrallah e Sedoc (2020) | SMRT | Não Utiliza | Não especifica | *BLEU, METEOR, ROUGE-L* e *Embedding Based Metrics* |
| Santos Júnior *et al.* (2021) | Não Utiliza | Não Utiliza | *Rasa* | Precisão, Revocação, *F1-Score* e Acurácia |
| Este trabalho | *FastText* | *WordNet* | *Rasa* | Precisão, Revocação, *F1-Score* e Acurácia |

Fonte: Elaborado pelo autor.

# 4 METODOLOGIA

Para alcançar o objetivo do trabalho é necessário realizar uma sequência de etapas, apresentadas na Figura 3 e as quais serão descritas posteriormente.

Figura 3 – Fluxo das etapas metodologicas.




Fonte: Elaborado pelo autor.

1. **Seleção de Tecnologias**. Esta etapa tem como objetivo selecionar as tecnologias utilizadas neste trabalho. Para a realização do treino e teste do modelo *NLU* criado, é utilizado o *framekwork Rasa*. A criação dos dados sintéticos de entrada *NLU* é realizada utilizando o *Chatette*, que é um programa *Python* que gera conjunto de dados de treinamento para o *RasaNLU*. Para a obtenção dos sinônimos é utilizado o *Magnitude* que é um pacote *Python* de código aberto que dispões de funções de similaridade que podem ser aplicadas aos *Word Embeddings*. Por fim, é selecionada também a tecnologia de recuperação de sinônimos através de uma base de conhecimento léxico *WordNet*.
2. **Modelagem e Implementação do modelo gerador de dados**. Esta etapa tem como objetivo estruturar e definir o escopo do modelo gerador de dados. A modelagem da aplicação é primeiro passo para definir os componentes, classes e métodos que o modelo gerador tem. A implementação do modelo tem como base a modelagem definida e as seleção das tecnologias selecionadas. A implementação é realizada utilizando a linguagem de programação *Python* e as principais bibliotecas voltadas para o processamento de dados e manuseio de texto.
3. **Implementação do *NLU***. A criação do modelo *NLU* é utilizada para a implementação do componente *Rasa NLU*. A partir dele é possível realizar o treino e teste do modelo, tendo em vista que a realização desses passos permite analisar a performance do modelo *NLU* gerado. Desta forma, nesta etapa já consideramos que os objetivos específicos 1 e 2 estão completos.
4. **Estudo de Caso**. Essa etapa tem como objetivo realizar experimentos utilizando dados reais, para que seja possível avaliar a funcionalidade do modelo gerador de dados. Estes experimento variam na utilização dos dados originais e dos dados com recuperação de

palavras similares com o *Word Embedding* ou *WordNet*. Os experimentos realizados nesta etapa utiliza da aplicação desenvolvida, assim como de todas as tecnologias selecionadas previamente.

5. **Avaliação dos resultados**. Para validar os resultados da objetivo deste trabalho é realizado a avaliação dos resultados do teste do modelo *NLU* utilizando a recuperação de palavras similares e os dados originais. Os resultados avaliados nesta etapa são provindos do estudo de caso realizado. Ao fim desta etapa, podemos considerar o objetivo específico 3 concluído.

# 5 SELEÇÃO DAS TECNOLOGIAS

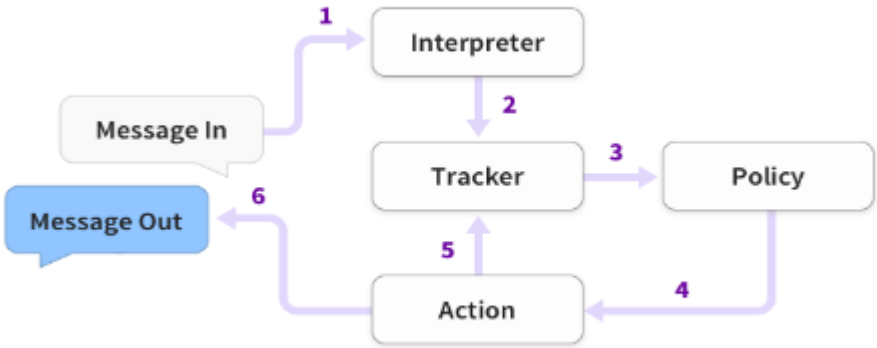
Este capítulo aborda as tecnologias selecionadas para a realização de alguns objetivos deste trabalho.

## 5.1 *Rasa*

Alguns *frameworks* de chatbots populares baseados em aprendizado de máquina são: *Microsoft Bot Framework* da *Microsoft*, *Dialogflow* do *Google*, *IBM Watson* da IBM, *Amazon Lex* da *Amazon* e *Rasa* da *Rasa Technologies Inc* (THAPA *et al.*, 2019).

O *Rasa* é um *framework* de código aberto usado para automatizar conversas baseadas em texto e voz. Ele é composto basicamente por dois componentes: *Rasa NLU* e *Rasa Core*. O *Rasa NLU* é responsável por entender as mensagens do usuário e extrair dados relevantes delas, assim como, pela classificação de intenções, extração de entidades e recuperação de respostas (THAPA *et al.*, 2019). Já o *Rasa Core* é responsável pelo fluxo da conversa entre o usuário e o chatbot (THAPA *et al.*, 2019).

Figura 4 – Fluxo do *Rasa Core*.




Fonte: Thapa *et al.* (2019).

Na Figura 4 é apresentada a arquitetura do fluxo do *Rasa Core* onde o *Menssage in* é responsável por receber a mensagem do usuário e enviar para o *Rasa NLU (Interpreter)*, o qual produz uma saída estruturada contendo o texto original, intenção e entidade, caso tenha; O *Tracker* recebe a saída do *Interpreter* e mantém o estado de conversação; O módulo *Policy* decide qual a próxima ação a ser executada; O *Action* é executado e seu *log* é mantido pelo *Tracker*; O *Menssage Out* fornece a resposta apropriada ao usuário (SHARMA; JOSHI, 2020).

Este trabalho dará suporte a geração dos dados do *NLU* para o *framework Rasa*. Assim como, utiliza do *framework Rasa* para treinar e testar o modelo *NLU* criado.

## 5.2 *Chatette*

*Chatette*[1] SimGus (2019) é um pacote do *Python* que gera um conjunto de dados de treinamento de modelos de arquivos para o *Rasa NLU*. O *Chatette* usa uma linguagem de domínio específico (DSL), permitindo assim definir modelos para gerar uma grande quantidade de frases, que são salvas no formato de entrada do *Rasa NLU*. Os dados que o *Chatette* usa e gera são carregados em arquivos. O arquivo de entrada contém os modelos e o de saída contém os exemplos gerados pelo *Chatette* que podem ser inseridos diretamente no *Rasa NLU*.

O arquivo de modelo é um arquivo que contém o texto em determinado formato. Nele será incluso declarações de unidades, as quais contém uma lista de regras de geração. As linhas que compõe o arquivo de modelo pode ser de quatro tipos:

1. Linhas vazias e comentários: A linha vazia são as linhas ausentes de caracteres. As linha de comentário começa com // seguido por palavras, e podem ser inseridas em qualquer lugar do documento. Ambas as linhas serão ignoradas pelo compilador.
2. Declaração de unidade: Esta linha não tem recuo e começa com os seguintes caracteres: ~, @ ou %. A escolha de qual caractere especial escolher vai depender do tipo de unidade que será definida. O primeiro caractere será seguido por um conjunto de caracteres entre colchetes e, em alguns casos, seguido por uma *strings* entre parênteses.
3. Definição de unidade: Estas linhas são o conteúdo das declarações de unidade e podem ser várias linhas. A indentação dessas linhas devem ser coerentes, ou seja, todas as linhas devem seguir um padrão de indentação. Cada linha refere-se a uma regra de geração da unidade a qual ela pertence.
4. Inclusão de outro arquivo: O conteúdo desta linha dirá ao analisador para incluir outro arquivo de modelo exatamente na linha que está. Ela começa com o simbolo | seguido pelo caminho do arquivo.

Na Figura 5 é apresentado um exemplo do arquivo modelo do *Chatette*. As linhas que estão indentadas na Figura 5 são o que chamamos de regras de geração. Essas regras são partes de uma sequência, podendo ser consideradas como sub-regras, capazes de gerar certas palavras. E a palavra proveniente da regra é uma concatenação dessas palavras. As sub-regras

[1] https://github.com/SimGus/Chatette

Figura 5 – Exemplo de arquivo de modelo Chatette.

```
~[synonym_which]
    qual
    quais
    o que é necessário

%[registration_form]
    ~[synonym_which] para realizar o cadastramento na @[ufc]?
    ~[synonym_which] documentos precisa no cadastro da @[ufc]?
    O cadastro na @[ufc] precisa de ~[synonym_which] documentos?
    A @[ufc] pede ~[synonym_which] documentos no cadastramento?

@[ufc]
    UFC
```

Fonte: Elaborado pelo autor.

podem ser separadas por caracteres especiais (~, @, % ou []) ou por espaços em branco. Existem três tipos de sub-regras, sendo elas:

1. Palavra: É uma palavra que sera gerada automaticamente.
2. Referência de unidades: É uma referência a outra definição de unidade. Isso é possível usando o caractere espacial seguido do nome da unidade entre colchetes.
3. Escolha: É uma lista de regras separadas por | entre colchetes ([|]) que dizem para o gerador escolher aleatoriamente uma das regras dentro do escopo e, assim, gerar a *string*.

Neste trabalho o *Chatette* será responsável por auxiliar na geração dos dados. Estes dados podem receber um acréscimo de novos valores através do uso das tecnologias de recuperação palavras similares. Entretanto, ainda é necessário que o usuário defina os arquivos de entrada. Assim, este trabalho visa auxiliar na construção dos *templates* e assim, automatizar essa etapa.

## 5.3 *Magnitude*

*Magnitude*[2] (PLASTICITYAI, 2020), é um pacote *Python* de código aberto que fornece um conjunto completo de recursos e um novo formato de arquivo de armazenamento vetorial que possibilita o uso de *embeddings* vetoriais de maneira rápida, eficiente e simples (PATEL *et al.*, 2018). O *Magnitude* pretende ser uma alternativa mais simples e rápida para as utilidades atuais de vetores de palavras, como *Gensim* (PATEL *et al.*, 2018).

Este trabalho utilizará algumas funções de similaridade do *Magnitude* para recuperação de possíveis sinônimos. Para realizar esta atividade o *Magnitude* utiliza arquivos do

[2] https://github.com/plasticityai/magnitude

tipo *".magnitude"* em vez de arquivos ".bin", ".txt", ".vec"e "hdf5"usados em *Word2Vec*, *GloVe*, *FastText* e *ELMo* (PATEL *et al.*, 2018). Como este trabalho utiliza o modelo de *Word Embedding FastText* foi necessário converter os modelos para o formato do *Magnitude*, utilizando um próprio comando de conversão do *Magnitude*. Desta forma, são utilizados os modelos *Word Embedding FastText CBOW* e *SkipGram* para que o *Magnitude* possa retornar os possíveis sinônimos.

## 5.4 *WordNet*

*Wordnet* é um recurso léxico organizado em uma estrutura hierárquica com base em *synsets* e nas características semânticas das palavras (CHAKRAVARTHI *et al.*, 2019). *Synset* é um conjunto (*set*) de sinônimos (*syn*), ou seja, um conjunto de palavras que podem ser substituídas entre si em algum contexto (PAIVA *et al.*, 2012). Originalmente o *WordNet* foi desenvolvido para língua inglesa pela Universidade de *Princeton*. No entanto, diversos projetos criaram um modelo do *WordNet* para sua língua e disponibilizaram este projeto no *Open Multilingual WordNet*. Tanto o *WordNet* quanto o *Open Multilingual WordNet* podem ser acessados pela biblioteca *Python NLTK*.

Este trabalho utilizará o *WordNet* na recuperação de sinônimos lematizados das palavras. Estes sinônimos são agregados no modelo *NLU* gerado.

# 6 ARQUITETURA DO GERADOR DE DADOS

A arquitetura proposta para realizar a geração dos dados tem como objetivo automatizar a geração dos sinônimos dos valores das entidades identificadas nos dados de entrada, e transformar esses dados para o formato *Chatette*. A arquitetura utiliza de *Word Embedding* e de uma base de conhecimento *WordNet* para gerar sinônimos e assim, aumentar quantidade de dados no arquivo de saída. Na Figura 6 é apresentado o diagrama de arquitetura da aplicação proposta. O diagrama apresenta a conexão da aplicação com as tecnologias utilizadas e com os dados de entrada e saída. Na camada dos dados de entrada são definidos os dados de entrada da aplicação, os *embeddings* utilizados e o dicionário de sinônimos utilizado pela biblioteca *WordNet*. Na camada da aplicação proposta são expostos o fluxo seguido pela aplicação e, a conexão das tecnologias de recuperação de sinônimos com o passo de obtenção de sinônimos. Na camada de dados sintéticos é apresentada a conexão entre os dados gerados pela aplicação e o *chatette*, o qual cria um novo arquivo com novos dados treinados no formato do *RasaNLU*. Por fim, os dados gerados na última camada é passado para a camada do *Rasa*, que irá receber esses dados e adicioná-los ao seu componente *NLU*.

Na próxima seção são apresentados a modelagem da aplicação, obtenção de sinônimos e criação dos dados no formato *chatette*.

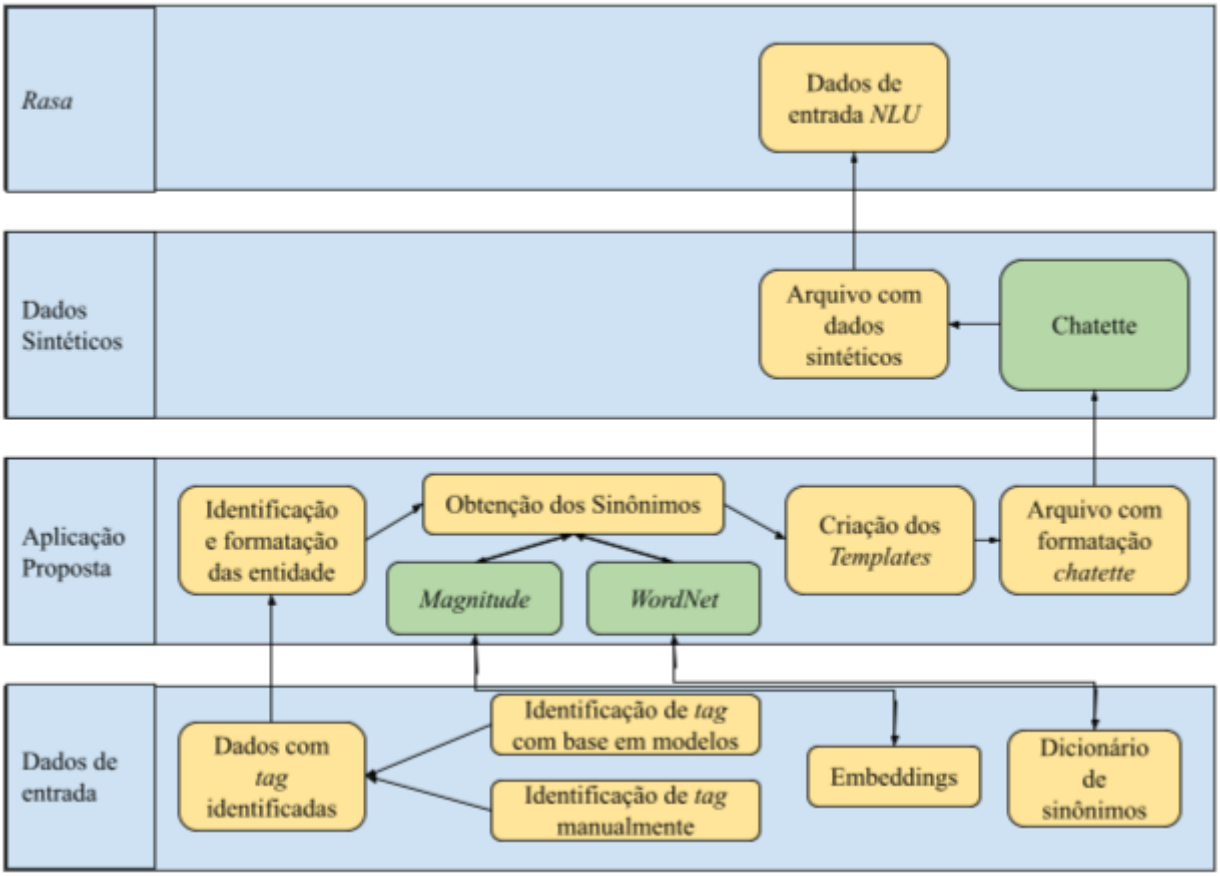
## 6.1 Modelagem

O modelo da aplicação é composto por cinco classes, sendo elas *Alaias*, *Entity*, *Intent*, *Synonyms*, *Template*.

A classe *Alaias* é responsável por adicionar ao arquivo um componente *alaias* com sua devida lista de valores. A classe *Entity* é responsável por extrair do conjunto de dados de entrada as entidades e seus valores e, adicioná-los ao arquivo de saída usando a formatação do *chatette*. Assim como, é possível adicionar novas entidades e seus respectivos valores. Como uma entidade pode ter vários valores, uma prática recomendável pelo *chatette* é segregar essa entidade em sub entidades. Assim, na classe *Entity*, foi utilizada a técnica de buscar por valores que tenham o mesmo radical para juntá-los em uma uma só entidade, garantindo assim, que os valores semelhantes fiquem na mesma entidade.

Na classe *Intent* é possível criar uma nova *intent* e seus respectivos valores, e escrever eles no arquivo *chatette*. A classe *Synonyms* é composta por três métodos diferentes de obtenção

Figura 6 – Diagrama de Arquitetura da aplicação proposta.



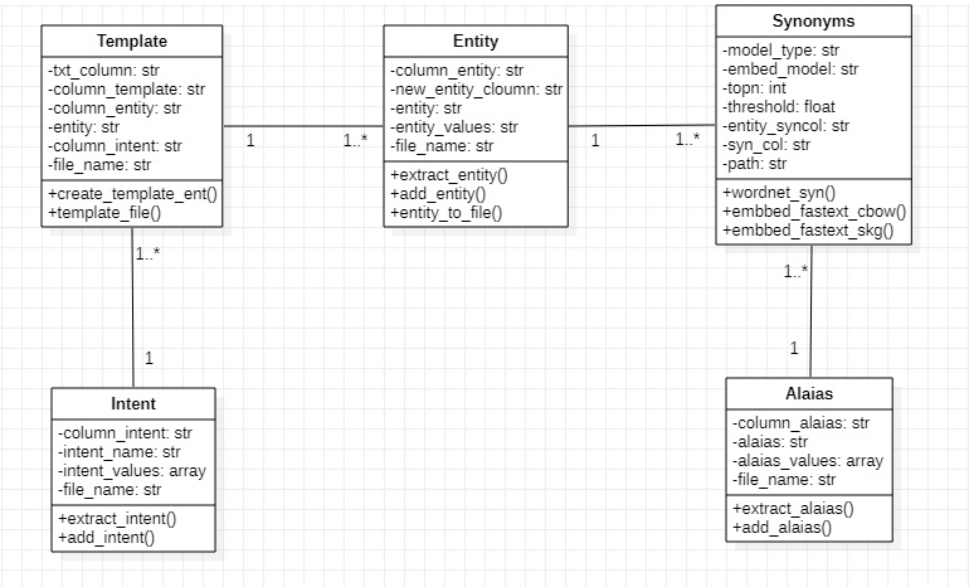

Fonte: Elaborado pelo autor.

dos sinônimos, sendo eles a obtenção através da base de conhecimentos *WordNet* e a obtenção através da utilização de modelos de *Word Embeddings FastText* do tipo *CBow* e *SkipGram*. Cada método que compõe a classe *Synonyms* recupera os sinônimos dos valores de uma determinada entidade. No caso da obtenção de sinônimos utilizando os modelos de *Word Embedding FastText*, é possível definir uma porcentagem de similaridade mínima para os valores da entidade e a quantidade de palavras que deseja retornar, esses parâmetros são chamados de *threshold* e *topn*, respectivamente.

Por fim, a classe *Template* é responsável por percorrer o conjunto de dados inicial, filtrar as sentenças pertencentes a cada intenção e identificar nelas, a presença de entidades. Como resultado, a classe *Template* retorna as intenções e suas sentenças com as entidades identificadas no texto. A classe *Template* também escreve no arquivo *chatette* o resultado do seu processamento. Na Figura 7 é apresentado o diagrama da arquitetura. Nela é exposto os componentes da arquitetura juntamente com as tecnologias que são utilizadas na aplicação.

A obtenção dos dados na formatação do *chatette* de forma automatizada deve seguir

Figura 7 – Diagrama de classe da arquitetura da aplicação proposta.




Fonte: Elaborado pelo autor.

uma sequência de passos. Inicialmente é necessário realizar uma chamada para classe *entity* que vai extrair e agrupar os valores daquela entidade que tem o radical igual, de forma a dividir melhor os valores pertencentes a ela. Por exemplo, na entidade *Sintoma* tem os valores *dor, febre, febril* então, a divisão ficaria *@[sintoma#dor]* com o valor *dor* e *@[sintoma#febre]* com os valores *febre* e *febril*. Esses valores são guardados com um dicionário que tem como chave o nome da entidade escrita no formato do *chatette* e como valor, os valores da dela.

Em seguida é possível obter os sinônimos desses valores, os quais serão armazenados junto com os demais valores do grupo. A obtenção dos sinônimos através dos modelos de *word embedding* necessitam de dois parâmetros obrigatórios, sendo eles o *topn*, que define uma quantidade de palavras similares a ser retornado, e o *threshold*, que delimita uma porcentagem miníma de similaridade que o valor retornado deve ter. Já a obtenção através do *WordNet* não exige nenhum parâmetro adicional, ele retornará todos os sinônimos encontrados de cada valor. Na Figura 8 é possível entender a diferença dos parâmetros necessários para cada tipo de recuperação de palavras similares.

Um vez extraídos os sinônimos, é realizada uma chamada para classe *Template*, a qual filtra nas sentenças pertencentes a cada intenção, valores de entidades contidas nelas. Ao encontrar algum valor de entidade, este valor é substituído pela chave da entidade a qual este valor pertence. Ao final, os textos são salvos com suas entidades marcadas nele.

Após extrair as entidades, obter os sinônimos e identificar as entidades no texto, é realizada a escrita desses dados em um arquivo *chatette*. É necessário realizar uma chamada

Figura 8 – Exemplo de chamada das funções de recuperação de palavras similares.

```
# Exemplo de chamada usando o WordNet
syn = Synonyms(data, '', '', '', '', 'dic_SINTOMA', 'syn_SINTOMA', path)
syn.wordnet_syn()

# Exemplo de chamada usando o Word Embedding FastText SkipGram
syn = Synonyms(data, '', '', 4, 0.7, 'dic_SINTOMA', 'syn_SINTOMA', path)
syn.embbed_fastext_skg()

# Exemplo de chamada usando o Word Embedding FastText CBOW
syn = Synonyms(data, '', '', 4, 0.7, 'dic_SINTOMA', 'syn_SINTOMA', path)
syn.embbed_fastext_cbow()
```

Fonte: Elaborado pelo autor.

para as classes *Entity* e *Template* passando o nome do arquivo com seu formato, como a exemplo *'dados_com_sinonimos_wordnet.chatette'*. Portanto, este é o arquivo final utilizado na geração de mais exemplos através da ferramenta *chatette*.

## 6.2 Geração dos dados sintéticos

O arquivo gerado pela aplicação[1] está no formato *chatette*, o qual é usado para gerar os dados com mais exemplos usando o comando interno do *chatette*. O arquivo de saída com o dados para treino gerado pelo o *Chatette* pode ser obtido em dois formatos, sendo eles o *JSON (.json)* e o *Markdown (.md)*. O *Rasa* aceita os formatos *JSON* e *YAML* como arquivo *NLU*. No entanto, como o *Rasa* utiliza o *YAML* como formato unificado e extensível de gerenciar todos os dados de treinamento, foi definido que o *NLU* seria neste formato. Para isso, o formato escolhido para gerar os dados de treino do *chatette* foi o *Markdown* o qual é convertido posteriormente para o formato *YAML* utilizando um *Script* desenvolvido para realizar esta conversão de formato. Este arquivo final formatado é o arquivo *NLU* que é utilizado pelo *Rasa*.

## 6.3 Implementação do *NLU*

A próxima etapa seguida é adicionar o arquivo *NLU* ao projeto *Rasa*. Após é dividido os dados *NLU* em treino e teste, com o propósito de separar uma parte dos dados para avaliar a qualidade do modelo treinado. Os dados são divididos em porcentagens de 80% para treino e 20% para teste. Ao dividir os dados em treino e teste é possível realizar o treinamento utilizando

[1] implementação disponível em: https://github.com/GabrielSBotelho/Automatic-NLU-Generator/

apenas os dados de treino, o qual ao final do treinamento, retorna um modelo treinado. O treino segue uma sequência de passos contidos em um arquivo de configuração do *Rasa*. Neste arquivo é passado como parâmetros a linguagem dos dados, a *pipeline* que define quais componentes são usados pelo modelo para realizar a previsão da *NLU* e as *policies* que define as políticas usadas para prever qual será a próxima ação do *NLU*. A *pipeline* utilizada neste trabalho teve como referência a *pipeline* utilizada em Santos Júnior *et al.* (2021), a qual será descrita a seguir. É utilizado o *SpacyNLP* para receber o modelo pré-treinado em português. Outro componente utilizado é o *SpacyTokenizer* para realizar a tokenização das palavras. Já o *SpacyFeatuarizer* é usado para incluir novos recursos aos dados. Por fim, é utilizado o *DIETClassifier* que é um modelo de classificação para treinamento do modelo *NLU*.

Definida as configurações e realizado o treino, o modelo deve ser capaz de reconhecer entradas pertencentes ao domínio ao qual os dados *NLU* refere-se. Para avaliar este modelo é utilizado os dados de teste, os quais servem como entrada para o modelo. Ao fim do teste é retornado os resultados contendo avaliações de desempenho do modelo na identificação das intenções e entidades. Foi realizado o experimento com os dados originais sem sinônimos, dados com obtenção de sinônimos da base de conhecimento *WordNet* e dados com obtenção de sinônimos através dos modelos de *Word Embedding FastText* [2] do tipo *CBOW* e *SkipGram*, ambos de 300 dimensões. No caso do experimento utilizando os *Word Embedding FastText* foram variados as quantidades de sinônimos retornados afim de analisar uma quantidade ideal que ajude na performance do modelo. Os valores foram 2, 4 e 6 sinônimos para o *CBOW* e *SkipGram*. Estes resultados são apresentados no próximo capítulo .

[2] modelos disponíveis em: http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc

# 7 ESTUDO DE CASO E RESULTADOS

Para alcançar os objetivos deste trabalho, foi utilizado o conjunto de dados obtidos em Santos Júnior *et al.* (2021). Estes dados são provenientes dos diálogos de profissionais da saúde através da Plataforma de Serviços do Coronavíros do estado do Ceará, Brasil. Em Santos Júnior *et al.* (2021) é realizada a limpeza dos dados removendo as frases duplicadas, informações pessoais, telefones, códigos postais, *emoticons*, *URL's* e locais. Ao final desse passo, sobrou um total de 33.000 frases. Seguindo, em Santos Júnior *et al.* (2021) foi realizada a extração das entidades utilizando a ferramenta do *SINTOMATIC* responsável por identificar sintomas no texto. Assim como, em Santos Júnior *et al.* (2021) é efetuado a classificação das intenções através da clusterização dos *embeddings*, tendo como resultado a identificação de cinco intenções. As intenções identificadas foram, *inform_medicine, request_inform, inform_symptoms, greeting, inform_diagnostic*. Portanto, o conjunto de dados disponibilizado por Santos Júnior *et al.* (2021) contém os textos classificados com a intenção a qual ele pertence, assim como, o conjunto de entidades identificadas em cada sentença. Para que fosse possível utilizar o conjunto de dados disponibilizado foram realizadas algumas edições, como seleção apenas da entidade 'Sintoma' e filtro de colunas que são essenciais para modelo gerador de dados. Na Figura 9 é apresentado um exemplo desses dados após a edição.

Figura 9 – Exemplo do conjunto de dados utilizado.

| intent | txt_clean | ents_SINTOMA |
|---|---|---|
| request_inform | consegue ir upa hoje noite? | [] |
| inform_diagnostic | tudo bem falta ar piorar deixe procurar atendi... | ['falta de ar'] |
| inform_diagnostic | piore hoje noite deixe procurar consulta manhã... | [] |
| request_inform | posso ajudalo algo? | [] |
| inform_diagnostic | sinais sintomas coronavírus principalmente res... | ['FALTA DE AR', 'febre', 'tosse', 'congestão n... |

Fonte: Elaborado pelo autor.

Na Figura 9 temos que são utilizadas 3 colunas principais, sendo elas a *'intent'* que indica a intenção que a sentença pertence. A coluna de *'txt_clean'* que contém todas as sentenças com seu texto tratado. E a coluna *'ents_SINTOMA'* que contém uma lista de valores da entidade 'Sintoma' que foram identificadas em cada sentença.

## 7.1 Geração dos dados sintéticos

A partir dos originais dados anotados foi possível utilizá-los como entrada para a aplicação desenvolvida. Desta forma, é iniciada a etapa de extração das entidades dele e transformação para o formato do *chatette*. Ao realizar a transformação é iniciado a obtenção de sinônimos usando os modelos de *Word Embedding* e o *WordNet*. Nessa etapa foi variado a quantidade de palavras similares retornadas em 2, 4 e 6 palavras para ambos modelos *Word Embedding FastText CBOW* e *SkipGram*. Também foi definida a porcentagem de similaridade mínima aceita em 70%. Após, é iniciada a identificação das entidades nas sentenças e por fim, é escrito no arquivo *chatette* as entidades e as intenções com seus respectivos valores formatados. Seguindo, o arquivo *chatette* obtido é executado de forma a criar novos exemplos de sentenças ao variar os valores das entidades contidas em cada sentença. Assim, ao final da execução é retornada um novo arquivo no formato *markdown* contendo todos os novos dados gerados. Para que esses dados sejam inseridos no *Rasa*, é necessário realizar a conversão do formato *markdown* para *yaml*. Esta conversão é realizada através de um *Script* desenvolvido, que recebe o arquivo *markdown* e adiciona os dados dele a um novo arquivo no formato *yaml*. Este arquivo final é o arquivo *NLU* que é usado pelo *Rasa* na realização do treino e teste do modelo.

## 7.2 Implementação e avaliação do módulo *NLU*

Finalizada a criação do arquivo *NLU* é criado um projeto *Rasa*, o qual recebe como entrada no componente *NLU* o arquivo gerado na etapa anterior. O arquivo *NLU* é dividido em 80% treino e 20% teste, assim, para o treinamento será usado os dados de treino e no teste os dados de teste. O treino é realizado seguindo a *pipeline* definida previamente. Foram treinados e avaliados 8 modelos *NLU* no total, sendo um com os dados originais e os demais com obtenção de palavras similares através do *Word Embedding* e *WordNet*. No caso dos modelos *NLU* com aumento de dados usando os *Word Embeddings FastText CBOW* e *SkipGram* foram variados o *topn* em 2, 4 e 6, gerando assim 3 modelos *NLU* para cada modelo *CBOW* e *SkipGram*.

Após concluir o treino é realizado o teste do modelo treinado. Ao final desse teste é gerado uma série de relatórios contendo informações sobre o desempenho do modelo na classificação das entidade e intenções. Neste trabalho as métricas analisadas na avaliação do desempenho do modelo são Precisão, Revocação e *F1-Score*.

No próximo capítulo são apresentado os resultados e comparações de todos os

experimentos realizados.

## 7.3 Resultados

Neste capítulo são apresentados os resultados dos experimentos descritos no Capítulo 7. Os dados exibidos neste capítulo tem o intuito de avaliar e realizar uma comparação do desempenho dos modelos *NLU* que realizaram a obtenção de sinônimos em relação a do modelo sem obtenção de sinônimos. Foram considerados as métricas de avaliação Precisão, Revocação e *F1-Score*. Assim como em Santos Júnior *et al.* (2021), este trabalho focou apenas na avaliação da entidade sintoma.

Tabela 1 – Tabela de comparação na classificação da entidade Sintoma

| **Entidade Sintoma** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métricas/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Precisão** | 0.74 | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** |
| **Revocação** | 0.66 | 0.79 | **0.89** | 0.88 | **0.89** | 0.81 | 0.88 | **0.89** |
| ***F1-Score*** | 0.70 | 0.88 | 0.93 | **0.94** | **0.94** | 0.89 | **0.94** | **0.94** |
| **Suporte** | 57 | 2.155 | 2.057 | 12.238 | 10.781 | 45.806 | 47.737 | 9358 |

Fonte: Elaborado pelo autor.

Na Tabela 1 são apresentados os resultados dos modelos na classificação da entidade sintoma. Nela, é possível analisar através das métricas que os dados com recuperação de sinônimos obtiveram uma performance melhor. O resultado dos dados com sinônimos comparado com os dados originais na métrica Precisão teve uma melhora de 25%. Já nas métricas de Revocação e *F1-Score* os modelos com sinônimo obtiveram uma melhora de até 23% e 24% respectivamente. A seguir é apresentado o resultado da classificação das intenções pelo modelo.

Na Tabela 2, é apresentada a comparação de desempenho dos modelos na classificação da intenção *inform_diagnostic*. As métricas de Precisão, Revocação e *F1-Score* mostram uma melhora de até 38%, 17% e 28% respectivamente nos modelos que receberam os sinônimos. Na Tabela 3 é avaliada o desempenho na classificação da intenção *inform_medicine*. Nela é possível notar uma melhora nas porcentagens das métricas as quais tem os dados com sinônimos. As métricas de Precisão, Revocação e *F1-Score* tiveram uma melhora de até 28%, 35% e 33% respectivamente.

Tabela 2 – Tabela de comparação na classificação da intenção *Inform_diagnostic*

| **Intenção *Inform_diagnostic*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métricas/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Precisão** | 0.62 | 0.99 | 0.98 | 0.99 | 0.99 | **1.0** | 0.99 | 0.99 |
| **Revocação** | 0.83 | 0.99 | 0.98 | 0.99 | **1.0** | 0.99 | 0.99 | **1.0** |
| ***F1-Score*** | 0.71 | **0.99** | 0.98 | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** |
| **Suporte** | 6 | 245 | 248 | 1.424 | 1.224 | 5.532 | 5.532 | 949 |

Fonte: Elaborado pelo autor.

Tabela 3 – Tabela de comparação na classificação da intenção *Inform_medicine*

| **Intenção *Inform_medicine*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métricas/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Precisão** | 0.60 | 0.68 | 0.60 | 0.69 | 0.67 | 0.71 | 0.76 | **0.88** |
| **Revocação** | 0.45 | 0.65 | 0.52 | **0.80** | 0.70 | 0.73 | 0.78 | **0.80** |
| ***F1-Score*** | 0.51 | 0.66 | 0.55 | 0.74 | 0.68 | 0.72 | 0.77 | **0.84** |
| **Suporte** | 20 | 23 | 23 | 31 | 30 | 38 | 38 | 40 |

Fonte: Elaborado pelo autor.

Na Tabela 4 exposta a comparação de desempenho na classificação da intenção *inform_symptoms*. Nesta intenção os dados gerados com acréscimo dos sinônimos também obtiveram um desempenho superior aos dados originais. Na Precisão, Revocação e *F1-Score* é possível destacar uma melhora de até 14%, 15% e 18% respectivamente. Na Tabela 5, é verificada a performance dos modelos na classificação da intenção *request_inform*. A Precisão, Revocação e *F1-Score* obtiveram uma melhora de desempenho de até 52%, 30% e 42% respectivamente nos modelos que teve obtenção de sinônimos.

Na Tabela 6 é exposto o desempenho dos modelos na classificação da intenção *Greeting*. Nesta intenção a maioria dos modelos com sinônimos ficaram abaixo da porcentagem em comparação aos dados originais. Na métrica Precisão o modelo *Word Embeddings FastText Cbow* com 4 sinônimos obteve uma melhora de 6%. Na métrica Revocação os modelos *Word Embeddings FastText Cbow* de 2 e 4 sinônimos e o *Word Embeddings FastText SkipGram* de 2 sinônimos obteve valores melhores de 2%, 16% e 7% respectivamente. Assim como o modelo *WordNet* que, no geral se manteve superior ou próximo do valor dos dados originais e

Tabela 4 – Tabela de comparação na classificação da intenção *Inform_symptoms*

| **Intenção *Inform_symptoms*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métricas/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Precisão** | 0.74 | 0.89 | 0.89 | 0.96 | 0.95 | **0.98** | 0.97 | 0.93 |
| **Revocação** | 0.84 | 0.94 | 0.93 | 0.98 | 0.95 | **0.99** | **0.99** | 0.97 |
| ***F1-Score*** | 0.80 | 0.92 | 0.91 | 0.97 | 0.95 | **0.98** | **0.98** | 0.95 |
| **Suporte** | 45 | 128 | 132 | 408 | 307 | 983 | 953 | 211 |

Fonte: Elaborado pelo autor.

Tabela 5 – Tabela de comparação na classificação da intenção *Request_inform*

| **Intenção *Request_inform*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métricas/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Precisão** | 0.47 | 0.85 | 0.68 | **0.99** | 0.91 | 0.96 | 0.98 | 0.95 |
| **Revocação** | 0.66 | 0.87 | 0.78 | 0.93 | 0.91 | **0.96** | **0.96** | **0.96** |
| ***F1-Score*** | 0.55 | 0.86 | 0.73 | 0.96 | 0.91 | 0.96 | **0.97** | 0.96 |
| **Suporte** | 15 | 33 | 33 | 111 | 86 | 284 | 284 | 141 |

Fonte: Elaborado pelo autor.

nesta métrica alcançou uma melhora de 15%. Na métrica *F1-Score* apenas os modelos *Word Embeddings FastText SkipGram* 2 sinônimos, *Word Embeddings FastText Cbow* 4 sinônimos e o *WordNet* obtiveram o valor acima da porcentagem dos dados originais, sendo eles 3%, 13% e 8% respectivamente.

Tabela 6 – Tabela de comparação na classificação da intenção *Greeting*

| **Intenção *Greeting*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métricas/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Precisão** | 0.66 | 0.61 | 0.63 | **0.73** | 0.50 | 0.56 | 0.6 | 0.65 |
| **Revocação** | 0.50 | 0.52 | 0.57 | **0.66** | 0.47 | 0.40 | 0.40 | 0.65 |
| ***F1-Score*** | 0.57 | 0.56 | 0.60 | **0.70** | 0.48 | 0.47 | 0.48 | 0.65 |
| **Suporte** | 20 | 21 | 21 | 21 | 21 | 22 | 22 | 21 |

Fonte: Elaborado pelo autor.

Na Tabela 7 é apresentada a comparação do desempenho dos modelos *NLU* na

Tabela 7 – Tabela de comparação da acurácia na classificação da intenções

| **Acurácia das Intenções** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Métrica/ Modelos *NLU*** | **Dados Originais Sem Sinônimos** | ***FastText CBOW Top2*** | ***FastText Skip-Gram Top2*** | ***FastText CBOW Top4*** | ***FastText Skip-Gram Top4*** | ***FastText CBOW Top6*** | ***FastText Skip-Gram Top6*** | ***WordNet*** |
| **Acurácia** | 0.67 | 0.92 | 0.90 | 0.97 | 0.97 | **0.98** | **0.98** | **0.98** |

Fonte: Elaborado pelo autor.

classificação das intenções. Nesta comparação é levada em consideração a acurácia ponderada de acordo com número de instâncias de cada intenção. A partir dos resultados apresentados na Tabela 7, é possível afirmar que os dados com obtenção de sinônimos obtiveram um desempenho superior em relação aos dados originais sem obtenção de sinônimos.

Tendo em vista que apenas a entidade Sintoma foi analisada neste trabalho é importante salientar que dado o contexto dos dados, a entidade foi encontrada em todas as intenções, dessa forma todas as intenções obtiveram o aumento de dados. Assim, após a análise dos resultados concluímos que a utilização dos sinônimos nos dados de entrada *NLU* garante uma performance consideravelmente melhor no desempenho do chatbot. Através dos ensaios é possível destacar um alto desempenho, na maioria dos casos, dos dados com sinônimos do *WordNet* e *Word Embeddings FastText CBOW* e *SkipGram* de 2, 4 e 6 sinônimos.

# 8 CONCLUSÃO

A fase inicial de criação dos dados de entrada para chatbots é um desafio, por ser uma atividade manual e ser caracterizada por ter uma pequena quantidade de dados iniciais. A utilização de ferramentas como o *Chatette* nesta fase inicial garante uma automatização na geração dos dados que são inseridos no *framework* de chatbot. Através da sua formatação o *chatette* consegue separar as principais partes do texto, como entidade, intenção e *alaias* e, com isso, gerar uma uma combinação de novos dados variando os valores das classes. No entanto, para que o *chatette* realize essa operação, os dados devem seguir uma formatação determinada por ele. Desta forma, este trabalho desenvolveu uma aplicação que recebe um conjunto de dados e retorna um arquivo na formatação do *chatette*. A aplicação também possibilita a geração de sinônimos dos valores de entidades e *alaias*, de forma a acrescentar novos dados aos dados originais. A obtenção dos sinônimos é garantida pela utilização de técnicas difundidas de recuperação de palavras similares através de *Word Embedding* e de bases de conhecimento léxico como o *WordNet*. Através dos testes realizados utilizando os dados originais e os dados com obtenção de quantidades variadas de sinônimos, é possível concluir que uma quantidade maior e variada de dados garante ao chatbot uma performance melhor.

Espera-se que essa aplicação sirva de apoio a desenvolvedores que ainda realizam a etapa de criar o arquivo *NLU* de forma manual. Essa etapa requer muito tempo e é suscetível a erros humanos. Deseja-se que ao ter o apoio de uma ferramenta que além automatizar esse processo, realiza através da obtenção de sinônimos o aumento de dados, agregue e facilite o trabalho do público-alvo. Neste trabalho foi apresentado que os modelos com recuperação de sinônimos com *Word Embedding FastText CBOW* e *SkipGram* variando entre 2, 4 e 6 sinônimos recuperados, garentem um desempenho superior em relação aos dados sem sinônimos. O mesmo se dá para o modelo com recuperação de sinônimos através do *WordNet*. Desta forma, espera-se que as análises realizadas sirvam como base para o público-alvo no momento de selecionar a técnica de recuperação de sinônimos.

Para trabalhos futuros, pode-se realizar experimentos com a geração de novos dados para do componente *alaias* da aplicação. Acredita-se que a adição desses novos dados garantam um aumento do desempenho dados *NLU* no *framework* de chatbot. Assim como, testar o desempenho desses modelos em outros *frameworks* de chatbot, de forma a analisar o comportamento desses dados em diferentes plataformas.

# REFERÊNCIAS

ABDELLATIF, A.; BADRAN, K.; COSTA, D.; SHIHAB, E. A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering. **IEEE Transactions on Software Engineering**, v. 5589, n. FEBRUARY, p. 1–18, 2021. ISSN 19393520.

CHAKRAVARTHI, B. R.; ARCAN, M.; MCCRAE, J. P. WordNet Gloss Translation for Under-resourced Languages using Multilingual Neural Machine Translation. **Proceedings of the Second Workshop on Multilingualism at the Intersection of Knowledge Bases and Machine Translation**, p. 1–7, 2019. Disponível em: https://www.aclweb.org/anthology/W19-7101.

CHEN, J.; AGBODIKE, O.; WANG, L. Memory-based deep neural attention (mDNA) for cognitive multi-turn response retrieval in task-oriented chatbots. **Applied Sciences (Switzerland)**, v. 10, n. 17, p. 1–11, 2020. ISSN 20763417.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

FORGUES, G.; PINEAU, J.; LARCHEVÊQUE, J.-M.; TREMBLAY, R. Bootstrapping dialog systems with word embeddings. **Nips, modern machine learning and natural language processing workshop**, v. 2, p. 168, 2014.

GAPANYUK; CHERNOBROVKIN, Y.; LEONTIEV, S.; LATKIN, A.; BELYANOVA, I.; MOROZENKOV, M.; OLEG. The hybrid chatbot system combining Q&A and knowledgebase approaches. **CEUR Workshop Proceedings**, v. 2268, p. 42–53, 2018. ISSN 16130073.

HARTMANN, N.; FONSECA, E.; SHULBY, C.; TREVISO, M.; RODRIGUES, J.; ALUISIO, S. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **arXiv preprint arXiv:1708.06025**, 2017.

HUSSAIN, S.; ATHULA, G. Extending a conventional chatbot knowledge base to external knowledge source and introducing user based sessions for diabetes education. **Proceedings - 32nd IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2018**, IEEE, v. 2018-Janua, p. 698–703, 2018.

KHAYRALLAH, H.; SEDOC, J. Smrt chatbots: Improving non-task-oriented dialog with simulated multiple reference training. **arXiv preprint arXiv:2011.00547**, 2020.

MALLIOS, S.; BOURBAKIS, N. A survey on human machine dialogue systems. **IISA 2016 - 7th International Conference on Information, Intelligence, Systems and Applications**, IEEE, 2016.

MASON, M. **3 types of business chatbots you can build**. 2019. Disponível em: https://www.ibm.com/blogs/watson/2017/12/3-types-of-business-chatbots-you-can-build/.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings**, p. 1–12, 2013.

NUGRAHA, H. S.; SUYANTO, S. Typographic-Based Data Augmentation to Improve a Question Retrieval in Short Dialogue System. **2019 2nd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2019**, IEEE, p. 44–49, 2019.

PAIVA, V. de; RADEMAKER, A.; MELO, G. de. Openwordnet-pt: An open Brazilian Wordnet for reasoning. In: **Proceedings of COLING 2012: Demonstration Papers**. Mumbai, India: The COLING 2012 Organizing Committee, 2012. p. 353–360. Published also as Techreport http://hdl.handle.net/10438/10274. Disponível em: http://www.aclweb.org/anthology/C12-3044.

PATEL, A.; SANDS, A.; CALLISON-BURCH, C.; APIDIANAKI, M. Magnitude: A fast, efficient universal vector embedding utility package. **arXiv preprint arXiv:1810.11190**, 2018.

PLASTICITYAI. **plasticityai/magnitude**. 2020. Disponível em: https://github.com/plasticityai/magnitude.

SANTOS JÚNIOR, V. O.; BRANCO, J. A. C.; OLIVEIRA, M. A. D.; SILVA, T. L. C. D.; CRUZ, L. A.; MAGALHAES, R. P. A natural language understanding model covid-19 based for chatbots. **2021 IEEE 21st International Conference on Bioinformatics and Bioengineering (BIBE)**, p. 1–7, 2021.

SETIAJI, B.; WIBOWO, F. W. Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling. **Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS**, IEEE, v. 0, p. 72–77, 2016. ISSN 21660670.

SHARMA, R. K.; JOSHI, M. An Analytical Study and Review of open source Chatbot framework, Rasa. **International Journal of Engineering Research and**, V9, n. 06, p. 1011–1014, 2020.

SIMGUS. **SimGus/Chatette**. 2019. Disponível em: https://github.com/SimGus/Chatette.

THAPA, S. *et al.* **Use of Word Embedding to Generate Similar Words and Misspellings for Training Purpose in Chatbot Development**. Tese (Doutorado), 2019.

VAKILI, A.; SHAKERY, A. Enriching Conversation Context in Retrieval-based Chatbots. **arXiv**, 2019. ISSN 23318422.

YSE, D. L. **Your Guide to Natural Language Processing (NLP)**. Towards Data Science, 2019. Disponível em: https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1.