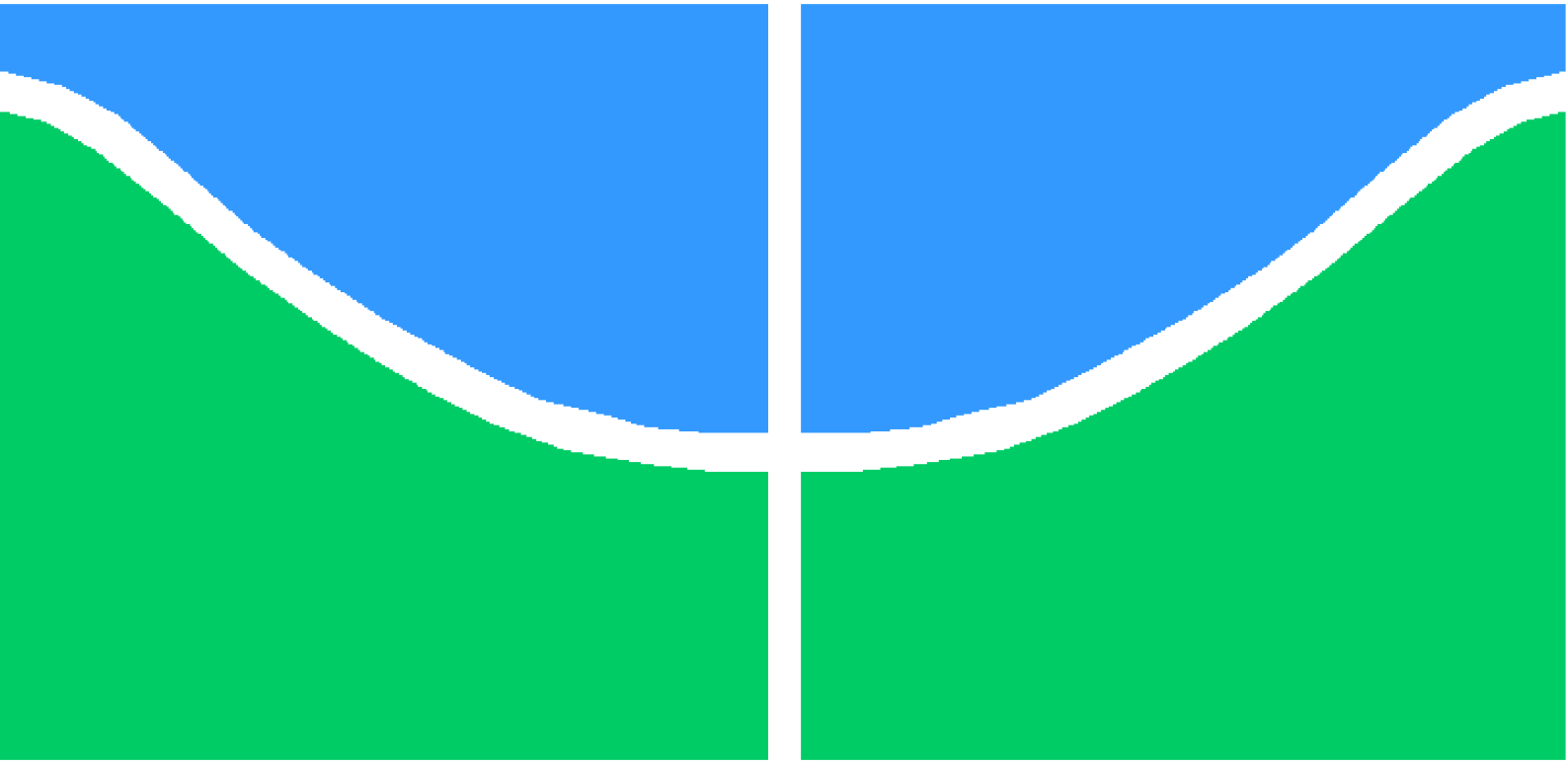


Universidade de Brasília - UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Engenharia de Software

Análise e Deliberação de Proposições Parlamentares Utilizando Sistemas Multiagentes e Aprendizado de Máquina

Autor: Douglas da Silva Monteles

Orientadora: Profª. Drª Milene Serrano

Brasília, DF

2025

Douglas da Silva Monteles

Análise e Deliberação de Proposições Parlamentares Utilizando Sistemas Multiagentes e Aprendizado de Máquina

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Orientadora: Profª. Drª Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília, DF

2025

Douglas da Silva Monteles

Análise e Deliberação de Proposições Parlamentares Utilizando Sistemas Multiagentes e Aprendizado de Máquina/ Douglas da Silva Monteles. – Brasília, DF, 2025-

148 p. : il. (algumas color.) ; 30 cm.

Orientadora: Profª. Drª Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE , 2025.

1. Sistemas Multiagentes. 2. Aprendizado de Máquina. I. Profª. Drª Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Análise e Deliberação de Proposições Parlamentares Utilizando Sistemas Multiagentes e Aprendizado de Máquina

CDU 02:141:005.6

Douglas da Silva Monteles

Análise e Deliberação de Proposições Parlamentares Utilizando Sistemas Multiagentes e Aprendizado de Máquina

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 20 de Fevereiro de 2025:

Profª. Drª Milene Serrano
Orientadora

Prof. Dr. Maurício Serrano
Coorientador

Prof. Dr. Renato Coral Sampaio
Convidado 1

Renan Welz Schadt
Convidado 2

Brasília, DF
2025

Agradecimentos

Primeiramente, gostaria de agradecer a Deus, que sempre me deu forças ao longo de todo o caminho percorrido até aqui, fortalecendo-me nos momentos de dificuldade, e estando sempre ao meu lado nos momentos de vitória.

Em segundo lugar, agradeço à minha família pelo amor incondicional e suporte emocional, que foram fundamentais em todos os momentos, especialmente à minha tia Marlene, que me proporcionou a oportunidade de sair do Maranhão para Brasília, o que me permitiu definir minha área de estudo.

Em especial, agradeço à minha orientadora, por sua orientação inestimável, paciência e apoio durante todo o processo de elaboração deste trabalho. Obrigado por acreditar em mim e me motivar a superar desafios.

Por fim, aos meus colegas e amigos, pelo incentivo constante e pelas discussões construtivas que enriqueceram este estudo.

Resumo

O processo de deliberação de proposições na Câmara dos Deputados é caracterizado por interações complexas entre parlamentares, partidos e outras influências políticas, refletindo a diversidade de interesses representados na Câmara. Este trabalho investiga a aplicação de sistemas multiagentes e aprendizado de máquina na análise de proposições parlamentares. Foram criados de agentes capazes de processar dados de votos, proposições e orientações de voto dos partidos, utilizando técnicas de redes neurais, como a *Long Short-Term Memory* (LSTM), para treinar os modelos com base em proposições que foram votadas entre os anos de 2008 à 2023, a fim prever comportamentos políticos e facilitar deliberações. Também apresenta a implementação de uma interface de usuário que permite selecionar uma votação e acompanhar em tempo real os agentes votando e comparar com os resultados reais. A pesquisa destaca a importância de ferramentas avançadas de processamento de linguagem natural e simulação de interações políticas, oferecendo uma visão embasada sobre como essas tecnologias podem ser aplicadas para melhorar a compreensão e a previsão de cenários legislativos.

Palavras-chaves: Inteligência Artificial. Sistema Multiagente. Aprendizado de Máquina. Aprendizado Profundo. Redes Neurais. Processamento de Linguagem Natural. Proposições Parlamentares.

Abstract

The deliberation process of propositions in the Chamber of Deputies is characterized by complex interactions among legislators, political parties, and other political influences, reflecting the diversity of interests represented in the Chamber. This study investigates the application of multi-agent systems and machine learning in the analysis of parliamentary propositions. Agents were developed to process voting data, propositions, and party voting guidelines, utilizing neural network techniques such as Long Short-Term Memory (LSTM) to train models based on propositions voted on between 2008 and 2023, with the goal of predicting political behavior and facilitating deliberations. It also presents the implementation of a user interface that allows selecting a vote, monitoring agents voting in real-time, and comparing the results with actual outcomes. The research highlights the importance of advanced natural language processing tools and the simulation of political interactions, providing an informed perspective on how these technologies can be applied to enhance the understanding and prediction of legislative scenarios.

Key-words: Artificial Intelligence. Multi-Agent System. Machine Learning. Deep Learning. Neural Networks. Natural Language Processing. Parliamentary Propositions.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CBOW	*Continuous Bag of Words*
CLI	*Command Line Interface*
CSV	*Comma-Separated Values*
DVCS	*Distributed Version Control System*
ESN	*Echo State Network*
FIPA	*Foundation for Intelligent Physical Agents*
GPU	*Graphics Processing Unit*
GPS	*Global Positioning System*
GRU	*Gated Recurrent Units*
GUI	*Graphical User Interface*
IAD	Inteligência Artificial Distribuída
JADE	*Java Agent DEvelopment Framework*
JDK	*Java Development Kit*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
LSTM	*Long Short-Term Memory*
MIT	*Massachusetts Institute of Technology*
PLN	Processamento de Linguagem Natural
RNA	Rede Neural Artificial
RNR	Rede Neural Recorrente
RBC	Raciocínio Baseado em Casos
SMA	Sistemas Multiagentes

TCC	Trabalho de Conclusão de Curso
TF-IDF	*Term Frequency-Inverse Document Frequency*
TI	Tecnologia da Informação
VCS	*Version Control System*
XML	*eXtensible Markup Language*

Lista de símbolos

θ	Letra grega minúscula theta
λ	Letra grega minúscula lambda
ϕ	Letra grega minúscula phi
Δ	Letra grega maiúscula Delta
$\mathcal{L}$	Letra L caligráfica

Sumário

1 Introdução

Este capítulo discorre sobre as informações que justificaram a escolha do tema desse Trabalho de Conclusão de Curso, bem como apresenta o viés de pesquisa e os objetivos pretendidos, que se encontram alcançados com a finalização desse trabalho. Para tanto, há uma Contextualização em relação ao domínio de interesse do trabalho, com destaque para o uso do paradigma de Sistemas Multiagentes no processamento de dados orientando-se por algoritmos de aprendizado. Para que seja compreensível o processamento de dados, deve-se ter em mente o conceito de "conhecimento" e a área de Gestão do Conhecimento. Será apresentada ainda a Questão de Pesquisa, respondida ao longo do trabalho, bem como a Justificativa para se contribuir nessa linha de pesquisa. Consonante a isso, serão descritos os Objetivos estabelecidos, os quais foram alcançados até o final da pesquisa. Por fim, tem-se a Organização da Monografia.

1.1 Contextualização

Em geral, o conhecimento pode ser explícito, geralmente registrado de alguma forma; ou implícito/tácito, que está apenas disponível na mente das pessoas e é difícil ou impossível de extrair. O conhecimento tácito é aquele armazenado no cérebro de uma pessoa, enquanto o conhecimento explícito é aquele contido em documentos ou outras formas de armazenamento que não seja o cérebro humano. Sendo assim, pode ser armazenado ou incorporado em instalações, documentos, bancos de dados, sites, e-mails, processos, serviços, sistemas, dentre outros. Ambos os tipos de conhecimento podem ser produzidos como resultado de interações ou inovações. Além disso, esses conhecimentos são mutuamente complementares. Sem o conhecimento tácito, será difícil, senão impossível, entender o conhecimento explícito (MALIKA, 2014).

A Gestão do Conhecimento é um processo de transformar várias informações disponíveis como conhecimentos explícitos e tácitos e combiná-las com pessoas utilizando sistemas baseados em bases de conhecimentos. Ela pode gerenciar a padronização do conhecimento, sendo benéfica para aquisição, utilização e inovação do conhecimento. Em resumo, as atividades básicas da Gestão do Conhecimento incluem adquirir, organizar, armazenar, compartilhar, aplicar e inovar o conhecimento (YUAN-YUAN; YONG-CHENG; HONG-MEI, 2011).

O processo de transformação de informações em conhecimentos gerenciáveis demanda uma série de atividades. Essas atividades devem ser realizadas de forma ordenada, começando pela coleta das informações (ou dados), passando por uma organização desses dados, no intuito de oferecer dados que sejam relevantes aos usuários, ou seja, que aten-

dam suas expectativas de alguma forma. A esse processo dá-se o nome de Processamento de Dados.

Para o Processamento de Dados, há necessidade do uso de Tecnologias da Informação (TI). Nesse contexto, tem-se a aplicação de computadores e equipamentos de telecomunicações para armazenar, recuperar, transmitir e manipular dados, frequentemente no contexto de uma empresa ou outra organização (MALIKA, 2014). O processamento de dados pode ser realizado de várias formas, dependendo dos insumos de TI. Além disso, há várias estratégias de solução para processar dados mais rapidamente, ou ainda com maior conformidade às necessidades dos usuários e aos recursos disponíveis. Dentre as principais estratégias, em Lobato et al. (2016), destacam-se:

- Processamento em Lote, no qual há envio de dados ao servidor, onde são organizados em lotes para um processamento posterior. Por exemplo, o processamento ocorrido nas leituras de consumo de água. Nesse caso, uma vez cumprida a leitura, o sistema gera um *backup* de todas as informações coletadas; armazena-as adequadamente, para depois gerar as faturas para os clientes;
- Processamento *Online*, no qual o processamento ocorre no momento exato da coleta dos dados. Por exemplo, quando se realiza o pagamento de uma compra, usando cartão de débito ou crédito, já ocorre o registro dessas transações junto à instituição financeira;
- Processamento *Offline*, no qual o processamento efetivo dos dados ocorre depois de todas as etapas concluídas, restando ser feita apenas a mudança de status final do processo. Por exemplo, quando se realiza algo em um dispositivo, desconectado da rede, permanecendo os dados armazenados no dispositivo até que seja possível estabelecer conexão para registro do status final em um sistema em nuvem ou hospedado em um servidor dedicado, e
- Processamento em tempo real, no qual o processamento ocorre ao longo de um período, continuamente, necessitando de atualizações constantes. Por exemplo, ao usar GPS, tem-se a atualização constante das rotas envolvidas, sendo necessário manter os rastros dos dados gerados ao longo do percurso realizado. Nesse caso, há registro em tempo real das ações executadas no sistema.

Ao avaliar esses diferentes processamentos, percebe-se que os mesmos demandam muito esforço para realizar as inerentes atividades associadas. Com a grande quantidade de dados que atualmente são coletados, torna-se uma tarefa muito árdua de ser feita - exclusivamente - por humanos. Portanto, deve-se optar por outras abordagens de solução para tratamento desses processamentos de dados usando apoio computacional. Novamente, existem várias opções, em diferentes paradigmas de programação. Por exemplo,

em Ponce (2018), o autor Lucas Ponce evidencia o uso do paradigma de Programação Paralela para processamento de dados massivos. Dentre outras abordagens, o presente trabalho foca no uso do paradigma de Sistemas Multiagentes.

Sistemas Multiagentes são definidos como um macrosistema, composto por agentes autônomos que buscam atingir objetivos individuais e interagem em um ambiente comum para resolver uma tarefa. Frequentemente, é visto como um paradigma para projetar aplicações complexas (BELGHACHE; GEORGÉ; GLEIZES, 2016).

Justamente por ser um paradigma que se orienta por uma abordagem híbrida, combinando vários paradigmas de programação, o autor do presente trabalho despertou seu interesse em conhecer melhor como essas entidades inteligentes e autônomas podem lidar com questões complexas, como ocorre no processamento de dados. Dentre os paradigmas inerentes no contexto de Sistemas Multiagentes, merecem menção: paradigma paralelo e concorrente, paradigma orientado a objetos e paradigma orientado a comportamentos. Detalhes sobre o paradigma de Sistemas Multiagentes serão tratados no REFERENCIAL TEÓRICO.

Optar pelo uso de abordagens mais aderentes aos desafios impostos na gestão do conhecimento e, portanto, no processamento dos dados associados, é algo praticado há muitos anos. Ao longo dos anos, pesquisadores de informações em mineração de dados e sistemas baseados em conhecimento recorreram a diferentes abordagens, com destaque para o uso de aprendizado multiestratégia. Essas abordagens facilitam a mineração de dados combinando o poder de sistemas de aprendizado conexionistas, como Rede Neural Artificial (RNA), aprendizado baseado em regras, e metodologias de computação distribuída. Estratégias de múltiplos agentes têm sido usadas na construção de muitos sistemas bem-sucedidos, nos quais um grupo de agentes trabalha coletivamente para realizar tarefas complexas (SHARMA; SHADABI, 2014).

Para esse trabalho, utilizou-se a base de dados abertos da Câmara dos Deputados Câmara dos Deputados (2024), que tem como objetivo oferecer, pela internet, um conjunto cada vez mais amplo de dados públicos sobre a Câmara dos Deputados. São dados puros, em formatos padronizados que os tornam mais adequados ao processamento por produtos de software de diversos tipos. Com isso, cidadãos e entidades da sociedade civil podem desenvolver aplicações computacionais que avaliam a atuação dos parlamentares, as votações da Casa, os gastos reembolsados com dinheiro público, dentre outros. Conferindo uma visão mais técnica sobre esses dados, a base de dados permite o acesso a dados puros, em formatos CSV, JSON e XML ou via API, que permite a aplicação de diversos filtros. Estes são formatos de serialização de dados, os quais possibilitam a troca de dados entre diferentes aplicações, plataformas ou sistemas de forma padronizada (NURSEITOV et al., 2009).

Os dados compartilhados na base de dados são referentes às categorias:

- **Blocos:** Nas atividades parlamentares, partidos podem se juntar em blocos partidários. Quando associados, os partidos passam a trabalhar como se fossem um "partido único", com um só líder e um conjunto de vice-líderes. Os blocos só podem existir até o fim da legislatura em que foram criados. Na legislatura seguinte, os mesmos partidos, se associados, formam um novo bloco;
- **Deputados:** Comportam os dados básicos sobre deputados que estiveram em exercício parlamentar em algum intervalo de tempo;
- **Eventos:** Compreendem informações básicas sobre eventos dos órgãos legislativos da Câmara, previstos ou já ocorridos, em um certo intervalo de tempo;
- **Frentes:** Representam agrupamentos oficiais de parlamentares entorno de um determinado tema ou proposta. As frentes existem até o fim da legislatura em que foram criadas, e podem ser recriadas a cada legislatura. Algumas delas são compostas por deputados e senadores;
- **Grupos:** Comportam equipes de interparlamentares, nas quais a Câmara teve representantes. As informações incluem nome, a resolução que criou o grupo e seu ano de publicação, e as mais recentes informações sobre situação, presidente e ofício de instalação;
- **Legislaturas:** Compreendem períodos de trabalhos parlamentares entre uma eleição e outra;
- **Partidos:** Comportam os dados básicos sobre os partidos políticos que têm ou já tiveram deputados na Câmara;
- **Proposições:** Representam informações básicas sobre projetos de lei, resoluções, medidas provisórias, emendas, pareceres e todos os outros tipos de proposições na Câmara;
- **Votações:** Compreendem as informações básicas sobre as votações ocorridas em eventos dos diversos órgãos da Câmara;
- **Órgãos:** Comportam as informações básicas sobre os órgãos legislativos e seus identificadores, tipos e descrições, e
- **Lideres:** Representam parlamentares que ocuparam cargos de liderança ao longo da legislatura.

Apesar dos diferentes tipos de dados compartilhados na base de dados, para esse trabalho, foram analisados somente os registros referentes às proposições que foram discutidas entre os anos de 2008 até 2023; as orientações dos partidos; os respectivos resultados

das votações realizadas, e as informações disponíveis sobre os deputados e partidos relacionadas ao tema votado. O fechamento do escopo foi para facilitar o tratamento e a exposição dos resultados. Além disso, essa escolha confere ao estudo a possibilidade de analisar a chance de aprovação ou não de uma dada proposição com base no seu tema e nos votos dos parlamentares.

Com estes dados, foi desenvolvido um software que, utilizando o paradigma de Sistemas Multiagentes em conjunto com Redes Neurais, consegue realizar uma análise das proposições discutidas, das orientações dos partidos quanto ao voto, dos resultados das votações realizadas, e criar agentes que conseguem deliberar[1] sobre as proposições apresentadas, visando prever o resultado de uma votação com base em resultados de votações de temas semelhantes e das características partidárias de cada deputado participante da votação.

1.2 Questão de Pesquisa

Ao longo deste trabalho, procurou-se responder à seguinte questão: **É possível criar um Sistema Multiagentes que consiga deliberar sobre as proposições apresentadas à Câmara dos Deputados, simulando o processo de votação com base em resultados anteriores sobre temas semelhantes?**

Para que essa questão fosse respondida, e diante de uma pesquisa prévia realizada com base na literatura especializada, foram utilizados algoritmos de aprendizado de máquina e outros recursos da Inteligência Artificial na modelagem e na implementação do Sistema Multiagentes.

1.3 Justificativa

O primeiro ato do processo de feitura de uma lei é a apresentação, à Casa Legislativa, de um projeto, de uma proposição legislativa. O processo de votação dar-se-á por meio dos sistemas de votação previstos no Regimento Interno da Câmara, o **simbólico** e o **nominal**. Pelo processo simbólico, que é o empregado na maior parte das votações, o presidente determina aos deputados favoráveis à aprovação do parecer do relator que permaneçam como estiverem (cabendo aos contrários manifestarem-se, em geral levantando a mão); declara o presidente, então, o resultado observado. No nominal, os membros são chamados, um a um, a declarar oralmente seus votos. Há três possibilidades de resultado para essa votação: aprovação integral, aprovação em parte, ou rejeição. Aprovado integralmente o parecer do relator, tal como apresentado à votação, passará a se constituir

[1] No contexto desse projeto, deliberação é processo pelo qual um agente de software toma decisões ou escolhe uma ação a ser executada com base em suas percepções, dados recebidos e objetivos.

no parecer final da comissão, encerrando a tramitação da proposição no âmbito do órgão (PACHECO, 2021).

O paradigma de Sistemas Multiagentes permite usar vários agentes de software, que atuam de forma colaborativa, para resolver problemas mais complicados, nos quais uma solução centralizada incorreria em uma complexidade muito alta (BELLIFEMINE; POGGI; RIMASSA, 2001a). Situações de maior complexidade ocorrem frequentemente devido a limitações de recursos (ex. servidores ou dispositivos com inadequada capacidade de processamento e memória, dentre outros). Nesses cenários, os agentes podem atuar em nome de um usuário, gerenciando informações quando necessário, de maneira distribuída, e respeitando o perfil do usuário registrado em suas configurações.

Os agentes exercem suas atividades de forma independente, pois são entidades autônomas. Entretanto, também possuem a capacidade de interagir entre si, lidando de forma conveniente com situações inerentemente distribuídas. No contexto de processamento de dados, essas habilidades dos agentes tornam-se relevantes, uma vez que os dados são comumente oriundos de diferentes fontes, em particular para o caso do domínio de aplicações de *e-health* (ou seja, aplicações no domínio da saúde, disponíveis em meio eletrônico). Por exemplo, um agente de consulta pode colaborar com um agente de diagnóstico para fornecer um retorno mais adequado ao paciente (SHARMA; SHADABI, 2014).

Considerando as colocações, justificou-se conduzir um estudo exploratório, cujo propósito foi o desenvolvimento de um Sistema Multiagentes, treinado com a base de dados das proposições já votadas, para que se adquira a capacidade de deliberar sobre as proposições apresentadas à Câmara dos Deputados, em um processo de votação nominal, pois este evidencia melhor a influencia de cada agente no resultado da votação, e com somente os resultados de Aprovação Integral ou Rejeição.

1.4 Objetivos

Os objetivos deste trabalho foram sub-divididos em Objetivo Geral e Objetivos Específicos. A seguir, consta a apresentação desses objetivos.

1.4.1 Objetivo Geral

Desenvolvimento de um sistema multiagentes capaz de deliberar, tomar uma decisão com base em informações disponíveis, sobre as proposições discutidas na Câmara dos Deputados e apresentar uma simulação da votação com base em temas votados anteriormente e nas características partidárias individuais de cada deputado.

Utilizou-se o paradigma de Sistemas Multiagentes para o processo de deliberação

e processamento dos dados. Também foi utilizado Aprendizado de Máquina para o treinamento de modelos capazes de analisar as proposições e as orientações de voto dos partidos dos deputados, a fim de extrair informações relevantes para o resultado da votação.

1.4.2 Objetivos Específicos

Tendo em vista o Objetivo Geral, foram considerados alguns Objetivos Específicos, conforme descritos a seguir:

- Estudo sobre a base de dados que contém as informações sobre as proposições discutidas na Câmara dos Deputados, os resultados das votações e as informações relacionadas aos deputados, como as orientações de voto dos seus respectivos partidos;
- Estudo sobre o paradigma de Sistemas Multiagentes;
- Estudo sobre processamento de dados e algoritmos associados (ex. Aprendizado de Máquina e Aprendizado Profundo com Redes Neurais Artificiais);
- Identificação de quais categorias referenciadas nas bases seriam de fato consumidas no processamento de dados: ora para o levantamento das proposições discutidas na Câmara dos Deputados, ora para deliberar sobre os assuntos e estabelecer a votação;
- Modelagem do Sistema Multiagentes capaz de processar os dados inerentes ao contexto e deliberar sobre;
- Disponibilização do Sistema Multiagentes para conhecimento dos interessados, e
- Análise do Sistema Multiagentes, considerando diferentes configurações de entrada.

Ressalta-se ainda que os estudos foram realizados orientando-se pelo método Pesquisa Bibliográfica e Pesquisa Documental; a modelagem e a disponibilização do Sistema Multiagentes foram orientadas às boas práticas da comunidade de software (i.e. notação de modelagem conhecida, quadro kanban para alinhamento das tarefas a serem realizadas e políticas de compartilhamento e contribuição esclarecidas sob a licença MIT), e a análise do Sistema Multiagentes foi orientada às etapas de aprendizado de máquina. Demais detalhes de cunho metodológico encontram-se apresentados na METODOLOGIA.

1.5 Organização da Monografia

O presente documento está estruturado da seguinte forma:

- **Capítulo 2 - Referencial Teórico:** confere uma visão geral sobre os principais conceitos e algoritmos que fundamentam o presente trabalho, com destaque para o paradigma de Sistemas Multiagentes e os algoritmos envolvidos no processamento de dados;
- **Capítulo 3 - Referencial Tecnológico:** apresenta as tecnologias, as ferramentas e os *frameworks* utilizados no desenvolvimento desse trabalho;
- **Capítulo 4 - Metodologia:** discorre sobre a forma como foi conduzido esse trabalho, descrevendo como ocorreram a Pesquisa Bibliográfica e a Pesquisa Documental nos estudos; como foram realizadas as etapas de cunho mais prático (ex. modelagem, implementação e disponibilização do Sistema Multiagentes), e como foi orientada a análise dos dados. Adicionalmente, é acordada uma visão temporal das atividades inerentes ao trabalho, com exposição de cronogramas;
- **Capítulo 5 - Sistema Multiagente de Simulação de Votação Parlamentar:** apresenta a proposta desse trabalho, com destaque para a origem da ideia; o escopo de atuação da pesquisa; as bases de dados; modelagem e arquitetura do Sistema Multiagentes; e algoritmos de aprendizado de máquina utilizados;
- **Capítulo 6 - Análise de Resultados:** descreve os resultados obtidos, sendo esses acompanhados por uma análise orientada à pesquisa-ação e às métricas específicas, e
- **Capítulo 7 - Conclusão:** apresenta as considerações finais do trabalho, com destaque aos objetivos cumpridos; resposta à questão de pesquisa; contribuições e limitações do trabalho, e ideias para trabalhos futuros.

2 Referencial Teórico

Este capítulo trata as principais fundamentações desse trabalho, abordando conceitos e teoria associada aos tópicos: Sistemas Multiagentes (Perfil do Usuário e Categorização); Aprendizado de Máquina (Processamento de Linguagem Natural); Redes Neurais Artificiais e vários subtópicos; *Framework Java Agent DEvelopment* (JADE) (Características, Arquitetura e Foundation for Intelligent Physical Agents). Por fim, têm-se as Considerações Finais do Capítulo.

2.1 Sistemas Multiagentes

Segundo Silveira (2006), ambientes de aplicações reais costumam ser compostos de um grande número de sub-problemas que, por sua vez, podem ainda ser divididos em sub-sub-problemas a fim de facilitar a análise e a implementação de soluções. Os Sistemas Multiagentes (SMA) são uma das estratégias da Inteligência Artificial Distribuída (IAD) que se mostra adequada para manipular problemas complexos. Este enfoque baseia-se no estudo e no desenvolvimento de agentes autônomos em um universo multiagente.

Para os SMA, o termo autônomo designa o fato de que os agentes têm uma existência própria, independente da existência de outros agentes. Usualmente, cada agente possui um conjunto de capacidades comportamentais que define sua competência; um conjunto de objetivos, e a autonomia necessária para utilizar suas capacidades comportamentais a fim de alcançar seus objetivos. Portanto, um agente é uma entidade computacional com um comportamento autônomo que lhe permite decidir suas próprias ações (SILVEIRA, 2006).

O comportamento apresentado pelos agentes para cumprir suas missões normalmente é explicado seguindo dois grandes modelos de funcionamento interno: agentes reativos e agentes cognitivos. Os **agentes reativos** têm comportamentos simples, uma vez que escolhem suas ações baseados unicamente nas percepções que possuem do ambiente. Nos **agentes cognitivos**, normalmente considera-se que os agentes possuem um estado mental e funcionam racionalmente, isto é, raciocinam para construir um plano de ações que leva a um objetivo pretendido (HÜBNER; SICHMAN, 2003).

2.1.1 Perfil do Usuário

O objetivo de uma arquitetura orientada a agentes de software é que os agentes consigam filtrar corretamente as informações dos usuários. Nesse sentido, é necessário que os agentes tenham conhecimentos, de alguma forma, do tipo de informação que esses

usuários estão interessados. Justamente através do perfil do usuário é que os agentes encontram os dados necessários para que possam desempenhar seus papéis corretamente (BALDASSIN; GUILHERME; MALTEMPI, 2002).

Tomando como base essa arquitetura, é utilizada uma abordagem dinâmica, em que é adotado o mecanismo de resposta do usuário como método de aprendizagem, tal como é visualizado nos agentes cognitivos. Entretanto, utilizando-se como entrada um conjunto de treinamento para construção do perfil inicial do usuário. Essa abordagem supõe que o usuário já possua um histórico de mensagens coletadas e separadas em categorias, seguindo critérios adotados pelo próprio usuário. Essas mensagens servirão como exemplos positivos ou negativos para construção de vetores protótipos de cada categoria. Dada uma nova mensagem, deve-se escolher em qual das categorias ela será classificada; ou ainda, se deve ser desconsiderada (BALDASSIN; GUILHERME; MALTEMPI, 2002).

No contexto do presente trabalho, os agentes possuem um perfil de usuário, que é construído através do processamento das informações disponíveis na base de dados relacionadas às proposições e as orientações de voto por parte dos partidos de cada um dos deputados.

2.1.2 Raciocínio Baseado em Casos

Na maioria das vezes, ao se procurar uma solução, ou uma explicação para um problema, apoia-se em situações passadas similares. Raciocínio Baseado em Casos (RBC) nada mais é do que isso. Sendo assim, trata-se de um método de soluções de problemas usando adaptações de soluções anteriores similares a estes problemas. Sistemas Baseados em Conhecimento (SBC) podem adaptar soluções anteriores para encontrar novas; usar casos anteriores para explicar novas situações e criticar novas soluções; raciocínios anteriores para interpretar uma nova situação; ou criar uma solução apropriada para um novo problema (CASTOLDI; SANTOS, 2002).

Ainda segundo Castoldi e Santos (2002), um sistema de RBC procura em uma base de casos (devidamente indexada), casos passados que se aplicam ao problema atual. Uma indexação coerente e o modo como os casos são representados facilitam na recuperação correta dos mesmos. O RBC possui três fases:

- definição do problema atual;
- recuperação dos casos, e
- adaptação ao contexto em análise.

É possível afirmar que RBC é uma metodologia tanto para raciocínio quanto para aprendizado. Para raciocínio, por utilizar os casos para auxiliar na solução ou na inter-

pretação de novos problemas. Para aprendizado, pela necessidade de armazenar as novas soluções ou interpretações geradas. O aprendizado torna o solucionador mais eficiente (CASTOLDI; SANTOS, 2002).

Para esse projeto, foi utilizada uma adaptação do modelo RBC, uma vez que, ao invés da base de dados com os casos indexados, foi utilizada uma base de casos cuja indexação foi substituída pela aplicação de técnicas de algoritmos de aprendizado profundo. Esse aprendizado fez uso dos dados disponíveis para ditar o julgamento apresentado pelos agentes.

2.1.3 Categorização

Baseando-se em Baldassin, Guilherme e Maltempi (2002), a categorização de textos consiste em associar um texto a uma ou mais categorias pré-definidas, baseando-se em seu conteúdo. Sendo assim, dada uma mensagem qualquer, é necessário escolher uma categoria, dentre as várias criadas pelo usuário e representadas pelo seu perfil, que melhor represente a ideia associada em seu conteúdo. Assim, caso uma mensagem tenha informação a respeito de futebol, por exemplo, essa seria associada a uma categoria que contenha informações sobre esportes.

No contexto desse trabalho, com interesse nas proposições parlamentares, as categorias tratadas pelos agentes já foram brevemente apresentadas no Capítulo 1 - Introdução, com destaque para Blocos, Deputados, Eventos, Frentes, Grupos, Legislaturas, Partidos, Proposições, Votações, Órgãos e Líderes.

Para que os interesses do usuário sejam refletidos dinamicamente em seu perfil, foram aplicadas técnicas de aprendizagem capazes de instruir o agente a se adaptar às mudanças de interesses de seus usuários (BALDASSIN; GUILHERME; MALTEMPI, 2002). Nesse contexto, cabe conhecer sobre Aprendizado de Máquina.

2.2 Aprendizado de Máquina

Em princípio, o termo “aprendizagem” refere-se ao conhecimento adquirido e retido na memória por experiência, esforço próprio, observação ou estudo (FERREIRA, 1999). Da mesma forma, no contexto da Ciência da Computação, aprendizagem também se refere à aquisição de conhecimento. Entretanto, isso ocorre na forma de dados interpretados, novas regras, ou mesmo na representação de uma experiência positiva ou negativa que é aproveitada em situações futuras (FONSECA, 2001).

Diante do exposto, e ainda com base em Fonseca (2001), apresenta-se como algo pertinente implementar agentes inteligentes com capacidades de auto-adaptação e aprendizagem, permitindo-lhes modificar seus comportamentos orientando-se pela experiência

adquirida e pela percepção de alterações no ambiente.

Para cada problema a ser resolvido, ou conhecimento a ser adquirido, existe um modelo de aprendizagem mais adequado. Logo, em uma comunidade multiagente, é perfeitamente possível coexistirem agentes adotando métodos e técnicas de aprendizagem diferentes. Dentre os diversos estudos e pesquisas sobre aprendizagem, algumas das técnicas mais conhecidas na IA são: Redes Neurais Artificiais (RNA) e Raciocínio Baseado em Casos (RBC) (OLIVEIRA et al., 2006). Mas, antes de adentrar nesses tópicos, cabe conhecer um pouco sobre Processamento em Linguagem Natural.

2.2.1 Processamento de Linguagem Natural

O processamento de linguagem natural pode ser definido como o campo da ciência da computação e linguística que estuda as interações entre seres humanos e máquinas, utilizando a linguagem natural. Isso ocorre, tanto na transformação de dados em algo que possa ser lido por seres humanos, quanto na conversão de amostras da linguagem natural em árvores de análise sintática, para que computadores possam as manipular mais facilmente (KUMAR, 2011).

Processamento de Linguagem Natural está voltado aos aspectos da linguagem natural, sendo: fonologia, morfologia e sintaxe, semântica e pragmática. Sendo assim, consiste no desenvolvimento computacional para a compreensão desta linguagem, fazendo com que a máquina tenha a capacidade de se comunicar, interpretar e processar de forma semelhante ao ser humano (JURAFSKY; MARTIN, 2019).

2.2.1.1 Análise de Sentimentos

A análise de sentimento é o campo de estudo que analisa a opinião de pessoas, assim como sentimentos, atitudes e emoções conectadas a um tópico ou produto. Historicamente, a análise de sentimento foi utilizada em variados níveis, começando pelo nível de documento, classificando o texto inteiro nele contido como tendo um sentimento positivo ou negativo; é utilizada também em nível de sentença, classificando se uma sentença em particular pode ser considerada como positiva, negativa ou neutra; e em nível de entidade e aspecto, realizando uma análise mais profunda para compreender se a opinião em si é positiva ou negativa, transformando texto não estruturado em dados estruturados (LIU, 2012).

2.2.1.2 Classificador de Texto

Sendo a classificação de texto a atribuição de uma categoria a um texto, como descrito anteriormente, um classificador de texto é um algoritmo que realiza esta tarefa de maneira automatizada. Dentro de possíveis soluções para a classificação de texto,

em aprendizado de máquinas, podem ser citados os classificadores baseados em Modelos de Máxima Entropia. Os modelos de máxima entropia são uma abordagem estatística amplamente usada em tarefas de processamento de linguagem natural e aprendizado de máquina. Baseiam-se no princípio da máxima entropia, que sugere que, ao construir um modelo probabilístico, deve-se escolher a distribuição que tenha a maior entropia (ou seja, a mais uniforme) possível, desde que obedeça às restrições impostas pelos dados (BARBOSA; MAAS; PEREIRA, 2019).

2.3 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um sistema projetado para modelar a maneira como o cérebro realiza uma tarefa particular, sendo normalmente implementada utilizando-se componentes eletrônicos; ou simulada por propagação em um computador digital. Para alcançarem bom desempenho, as redes neurais empregam uma interligação maciça de células computacionais simples, denominadas de "neurônios" ou unidades de processamento (HAYKIN, 2001).

As RNAs são comumente utilizadas na resolução de problemas complexos, onde o comportamento das variáveis não é rigorosamente conhecido. Uma de suas principais características é a capacidade de aprender por meio de exemplos e de generalizar a informação aprendida, gerando um modelo não-linear, tornando sua aplicação na análise espacial bastante eficiente (SPöRL; CASTRO; LUCHIARI, 2011).

A grande quantidade de modelos existentes exige uma análise de suas principais propriedades e diferenças em detrimento de uma análise caso a caso mais detalhada. O estudo das principais propriedades das redes neurais permite compreender melhor as vantagens e/ou inconvenientes da escolha de um modelo em detrimento de outro. Ressalta-se que não há apenas uma maneira de classificar todos os modelos. Entretanto, de modo geral, devem ser considerados grupos de atributos, tais como: tipo de aprendizado, arquitetura de interconexões, forma interna de representação das informações, tipo de aplicação da rede, entre outros (OSÓRIO; BITTENCOURT; OSÓRIO, 2000).

Há várias formas de se implementar Redes Neurais Artificiais. Dentre as mais conhecidas, cabe menção às descritas nas seções 2.3.1 e 2.3.2.

2.3.1 Redes Neurais de Propagação Direta

São redes neurais que reagem a mudanças no seu ambiente, incluindo *perceptrons*, de acordo com Rosenblatt (1958), e redes de função de base radial, segundo The Asimov Institute (Acesso em: 10 de abr. 2024), transformam padrões de entrada em saída. Elas são o arquétipo da rede neural, tendo camadas que consistem em nós de entrada, ocultos ou de saída. Os nós estão conectados entre camadas adjacentes, que podem ser totalmente

conectadas (cada neurônio de uma camada com cada neurônio em outra camada). A rede mínima tem duas células de entrada e uma célula de saída, que podem ser usadas para modelar portas lógicas, por exemplo.

Retropropagação é um algoritmo de aprendizado comum, onde a rede é mostrada como pares de entrada e saída esperadas, e a força das conexões entre os nós é atualizada com base em uma métrica orientada ao sucesso do modelo em prever algo. Teoricamente, dadas infinitas unidades em uma única camada oculta não linear, qualquer relação entre padrões de entrada e saída pode ser aprendida. No entanto, ter várias camadas ocultas (criando assim uma rede profunda) pode, na prática, levar a um processo de aprendizado mais eficiente (LEIJNEN; VEEN, 2020).

2.3.1.1 Redes Convolucionais

As redes convolucionais, segundo LeCun et al. (1998), são arquiteturas de aprendizado profundo que geralmente contêm camadas convolucionais e de *pooling*, usadas para varredura aproximada de padrões que frequentemente são correlacionados espacialmente. Assim, são úteis para processamento de imagens, mas podem ser aplicadas a outras modalidades de dados também. Em outras palavras, trata-se de um algoritmo que permite, a partir de uma imagem de entrada, atribuir relevância a vários "objetos" da imagem, diferenciando-os. Atribuir relevância significa, por exemplo, conferir pesos ou tendências/vieses que podem ser aprendidos.

As camadas deconvolucionais, segundo Zeiler et al. (2010), produzem resultados inversos e, portanto, podem ser usadas para geração de imagens. Redes gráficas inversas convolucionais profundas, de acordo com Kulkarni et al. (2015), são outro tipo que pode ser usado para (parcialmente) gerar imagens, sendo semelhante aos *autoencoders* variacionais, mas equipadas com nós convolucionais para as camadas de codificação e decodificação (LEIJNEN; VEEN, 2020).

2.3.1.2 *Autoencoders*

Autoencoders comprimem (codificam) e regeneram (decodificam) as informações, transformando-as em camadas ocultas menores (i.e. com menor número de nós na rede neural) com camadas simétricas ao redor. A semelhança entre entrada e saída pode ser usada como uma medida de sucesso para a compressão (LEIJNEN; VEEN, 2020).

Os *variational autoencoders*, segundo Leijnen e Veen (2020), compartilham uma arquitetura semelhante. Entretanto, aprendem orientando-se por uma distribuição de probabilidade aproximada dos padrões de entrada com base na inferência Bayesiana e na modelagem de relações causais. Sendo assim, é um modelo baseado em inferência. Na inferência Bayesiana, em resumo, o aprendizado se dá, paradigmaticamente, pela escolha da hipótese de máxima verossimilhança. Uma melhoria significativa nas capacidades de

representação dos autoencoders foi alcançada pelo modelo de Autoencoders Variacionais (VAE), de acordo com Kingma e Welling (2013). Seguindo a Inferência Variacional de Bayes (VB), segundo Bishop (2006), os VAE são modelos generativos que tentam descrever a geração de dados através de uma distribuição probabilística.

Especificamente, dado um conjunto de dados observados $X = \{x_i\}_{i=1}^{N}$ de N amostras i.i.d, assumi-se um modelo generativo para cada dado x_i, condicionado a uma variável latente aleatória não observada z_i, onde θ são os parâmetros que governam a distribuição generativa. Este modelo generativo também é equivalente a um decodificador probabilístico. De forma simétrica, assumi-se uma distribuição posterior aproximada sobre a variável latente z_i dado um dado x_i, denotado por reconhecimento, que é equivalente a um codificador probabilístico e é governado pelos parâmetros ϕ. Finalmente, assumi-se uma distribuição a priori para as variáveis latentes z_i, denotada por $p_\theta(z_i)$. Os parâmetros θ e ϕ são desconhecidos, e precisam ser aprendidos a partir dos dados. As variáveis latentes observadas z_i podem ser interpretadas como um código dado pelo modelo de reconhecimento $q_\phi(z \mid x)$.

O logaritmo da verossimilhança marginal é expresso como uma soma sobre os pontos de dados individuais, conforme Equação (2.1):

$$\log p_\theta(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N} \log p_\theta(x_i), \tag{2.1}$$

e cada ponto pode ser reescrito, como consta na Equação (2.2):

$$\log p_\theta(x_i) = D_{KL}\left(q_\phi(z|x_i) \| p_\theta(z|x_i)\right) + \mathcal{L}(\theta, \phi; x_i) \tag{2.2}$$

Os *denoising autoencoders* são outro tipo de *autoencoder*, onde os dados de entrada são processados através de um filtro de ruído aleatório (por exemplo, tornando uma imagem granulada). A saída ainda é comparada à imagem de entrada original, para que a rede aprenda a ignorar algumas das características detalhadas que não são causalmente relevantes (LEIJNEN; VEEN, 2020).

Finalmente, os *sparse autoencoders* fazem quase o inverso dos *autoencoders* anteriores, projetando informações para uma camada oculta maior, em vez de menor. Isso permite que a rede se concentre em características de menor nível de granularidade (i.e. mais específicas) ao comprimir e reconstruir os dados de entrada. Para evitar que as informações sejam copiadas perfeitamente entre as camadas, um filtro é usado para o erro que está sendo retropropagado (LEIJNEN; VEEN, 2020). Deve-se adicionar uma penalidade, conhecida como penalidade de esparsidade, ao critério de treinamento do modelo. O erro de reconstrução é dado pela Equação (2.3):

$$\arg min_{A,B} \mathrm{E}[\Delta(x, B \circ A(x))] + \lambda \sum_i |a_i|, \tag{2.3}$$

onde $\arg min_{A,B}$ indica a procura dos valores dos parâmetros A e B que minimizam a função de custo; $\Delta(x, B \circ A(x))$ representa uma medida de diferença ou erro entre o dado de entrada x e sua reconstrução através da combinação das funções A e B; $\mathrm{E}[\cdot]$ denota o valor esperado sobre todas as possíveis entradas x, indicando o interesse na média do erro de reconstrução para todas as amostras possíveis do conjunto de dados; λ é um parâmetro de regularização que controla o peso da penalidade de esparcidade em relação ao erro de reconstrução, e $\sum_i |a_i|$ calcula a soma das magnitudes das ativações a_i em todas as camadas ocultas do modelo (BANK; KOENIGSTEIN; GIRYES, 2023).

2.3.1.3 Redes de Atenção

As Redes de Atenção, de acordo com Jaderberg, Simonyan e Zisserman (2015), representam uma classe de redes em vez de uma arquitetura específica. Elas empregam um mecanismo de atenção para evitar a perda de informações, armazenando separadamente os estados anteriores da rede e alternando a atenção entre esses estados. Esse contexto pode ser visualizado, proporcionando *insights* interessantes sobre as correlações entre as características de entrada e as previsões (LEIJNEN; VEEN, 2020).

Por envolver uma classe de redes neurais de processamento, é possível correlacionar as informações e indicar a alternativa com a sequência correta de processamento. O uso de mecanismos de atenção permite detectar estímulos inesperados, ou que acontecem na mudança do foco atencional, simulando o que acontece em um cérebro humano. Tem sido utilizada com sucesso em contexto variados, tais como no reconhecimento de placas de trânsito (RODRIGUES, 2002).

2.3.2 Redes Neurais Recorrentes

Redes recorrentes são redes de propagação com conexões dentro das camadas. Portanto, elas possuem estado, e o tempo e a ordem em que o *input* é estruturado são importantes. Isso permite que as redes recorrentes encontrem estruturas no tempo (ELMAN, 1990). Elas também podem ser usadas com modalidades de dados independentes do tempo, tal como ocorre com imagens, as quais podem ser representadas como uma sequência (por exemplo, de pixels). Treinar essas redes pode resultar em gradientes desaparecendo (ou explodindo), onde, dependendo das funções de ativação usadas, a informação se perde (ou é amplificada) ao longo do tempo, de forma semelhante ao que acontece com redes de propagação muito profundas, que podem perder informações em profundidade (LEIJNEN; VEEN, 2020).

Redes neurais recorrentes são bem adequadas para tarefas que envolvam o armazenamento e a atualização de informações de contexto, isso porque possuem um estado interno que pode representar essas informações de contexto. Os ciclos no grafo de uma rede recorrente permitem que ela mantenha informações sobre entradas passadas por um período de tempo que não é fixado a priori, mas que depende de seus pesos e dos dados de entrada. Em contraste, redes estáticas (ou seja, sem conexão recorrente), mesmo que incluam atrasos, têm uma resposta a impulso finita e não podem armazenar um bit de informação por um tempo indefinido (BENGIO; SIMARD; FRASCONI, 1994).

Uma RNR oferece um método no qual as saídas das previsões anteriores são usadas como uma entrada adicional para a próxima iteração Nashold e Krishnan (2020). No entanto, a RNR mostra-se ineficaz em longos intervalos, pois algoritmos de aprendizagem baseados em gradiente enfrentam um problema cada vez mais difícil à medida que aumenta a duração das dependências a serem capturadas, resultando no problema de desaparecimento ou explosão do gradiente, o que causa falha em aprender de forma eficaz com os dados de treinamento, resultando em um ajuste insuficiente (BENGIO; SIMARD; FRASCONI, 1994).

2.3.2.1 Memória de Curto e Longo Prazo

A Memória de Curto e Longo Prazo (LSTMs) fornece uma solução para os problemas de gradientes desaparecendo e explodindo, introduzindo *gates* (portas) e células de memória definidas explicitamente. Cada nó possui uma célula de memória e três *gates*: *input* (entrada), *output* (saída) e *forget* (esquecimento). A função dessas "portas" é proteger a perda de informação, permitindo ou bloqueando seu fluxo. A porta de entrada determina quanto da informação da camada anterior é armazenada na célula. A porta de saída determina o que a próxima camada deve saber sobre o estado desta célula. A porta de esquecimento evita que novas informações sejam ignoradas. As Unidades Recorrentes com Portas (GRUs) são LSTMs com um conjunto diferente de portas, tornando-as mais rápidas, mas menos expressivas (LEIJNEN; VEEN, 2020).

As LSTM são projetadas explicitamente para evitar o problema de dependência de longo prazo. Seu comportamento padrão é lembrar informações por longos períodos (OLAH, 2015).

A Figura 1 apresenta a arquitetura de um bloco de LSTM. A arquitetura consiste em um conjunto de sub-redes conectadas recorrentemente, conhecidas como blocos de memória. A ideia por trás do bloco de memória é manter seu estado ao longo do tempo e regular o fluxo de informações através de unidades de portão não lineares (HOUDT; MOSQUERA; NÁPOLES, 2020).

- **Entrada do bloco**: Esta etapa é dedicada à atualização do componente de entrada

do bloco, que combina a entrada atual $x(t)$ e a saída daquela unidade LSTM $y(t-1)$ na última iteração;

- **Porta de entrada**: Nesta etapa, atualiza-se a porta de entrada que combina a entrada atual $x(t)$; a saída daquela unidade LSTM $y(t-1)$, e o valor da célula $c(t-1)$ na última iteração;

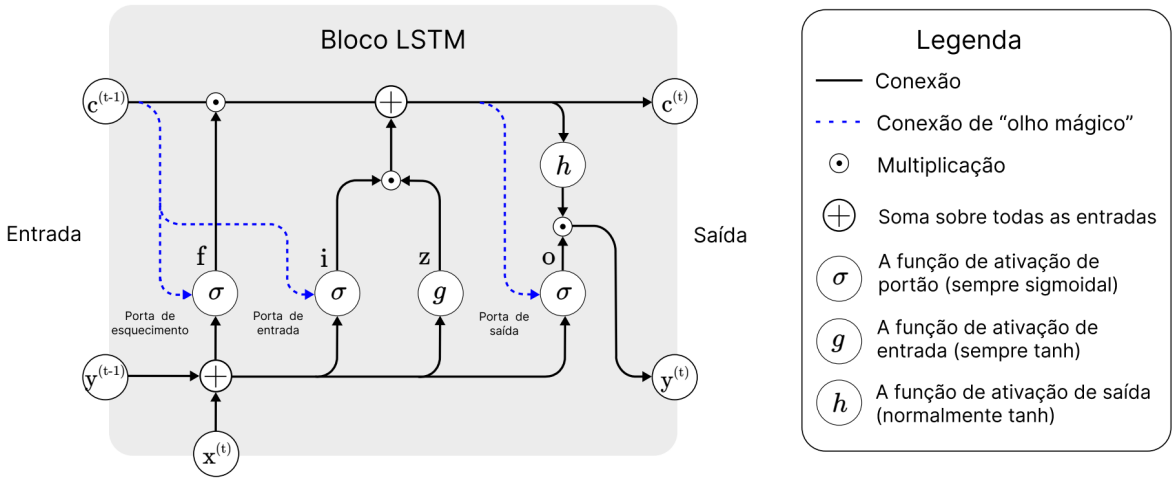
- **Porta de esquecimento**: Nesta etapa, a unidade determina quais informações devem ser removidas dos seus estados de célula anteriores $c(t-1)$. Portanto, os valores de ativação $f(t)$ das portas de esquecimento no instante de tempo t são calculados com base na entrada atual $x(t)$; nas saídas $y(t-1)$, e no estado $c(t-1)$ das células de memória no instante de tempo anterior $(t-1)$, nas conexões de espiada (conexão de "olho mágico") e nos termos de viés b_f das portas de esquecimento;

- **Célula**: Esta etapa calcula o valor da célula, que combina a entrada do bloco $z(t)$, os valores da porta de entrada $i(t)$ e da porta de esquecimento $f(t)$, com o valor anterior da célula;

- **Porta de saída**: Esta etapa calcula a porta de saída, que combina a entrada atual $x(t)$; a saída daquela unidade LSTM $y(t-1)$, e o valor da célula $c(t-1)$ na última iteração, e

- **Saída do bloco**: Combina o valor atual da célula $c(t)$ com o valor atual da porta de saída.

Figura 1 – Arquitetura de um bloco LSTM



Fonte: Houdt, Mosquera e Nápoles (2020) - Tradução (Autor).

2.4 *Framework Java Agent DEvelopment* (JADE)

O *Java Agent Development Framework* (JADE) é um *framework* Java para o desenvolvimento de aplicações distribuídas multiagentes. Mesmo sendo um ferramental para o trabalho, o intuito dessa introdução ao *framework* no Referencial Teórico do trabalho é esclarecer sobre o modelo conceitual que orienta a arquitetura de agentes de software nessa plataforma de desenvolvimento (KUMAR; KUMAR, 2014).

Trata-se de um *middleware* de agentes que fornece um conjunto de serviços disponíveis e fáceis de usar, além de várias ferramentas gráficas para depuração e teste. Um dos principais objetivos da plataforma é oferecer apoio à interoperabilidade, seguindo rigorosamente as especificações FIPA (*Foundation for Intelligent Physical Agents*)[1] referentes à arquitetura da plataforma e à infraestrutura de comunicação. Além disso, o JADE é muito flexível e adaptado para ser usado em dispositivos com recursos limitados, como *smartphones* (KUMAR; KUMAR, 2014).

Detalhes sobre o JADE, no que compreende suas principais características, sua arquitetura e seus protocolos padronizados pela FIPA, constam apresentados a seguir.

2.4.1 Características

O JADE é um *framework* de software, independente de aplicação específica, capaz de prover funcionalidades de camada de *middleware*. Ele possui uma infra-estrutura bastante flexível para o desenvolvimento de aplicações, na qual o elemento de abstração é um agente de software. Para isso, ele oferece apoio ao ciclo de vida e à lógica do núcleo de agentes em si, além de uma variedade de ferramentas gráficas que favorecem o desenvolvimento (TEIXEIRA, 2010).

O JADE baseia-se nos seguintes princípios, segundo Bellifemine, Poggi e Rimassa (2001b):

- Interoperabilidade: está em conformidade com as especificações FIPA, possibilitando que os agentes possam se comunicar com outros que não estejam atuando em ambiente de execução JADE;

- Uniformidade e Portabilidade: provê aplicações com um conjunto homogêneo de APIs (*Application Programming Interfaces*), sendo essas independentes da estrutura subjacente e da versão Java. O ambiente de execução JADE provê as mesmas APIs tanto para J2SE como para J2SE e J2ME e, em teoria, o desenvolvedor pode escolher o ambiente de execução Java em tempo de compilação;

[1] Disponível em: <http://fipa.org>. Acesso em: 12 fev 2025.

- Facilidade de Uso: a complexidade do *middleware* é ocultada por trás de um simples e intuitivo conjunto de APIs, e

- Filosofia *Pas-you-go*: programadores não são obrigados a usar todas as características providas pelo *middleware*. Aquelas que não serão utilizadas não requerem conhecimento algum por parte do programador, tampouco adiciona qualquer sobrecarga computacional.

2.4.2 Arquitetura do JADE

O JADE possui vários elementos que o compõe. O *framework* é composto por *containers* de agentes, que possivelmente podem estar distribuídos pela rede. Esses *containers* devem ser habitados por agentes, correspondendo ao processo Java que provê o ambiente de execução JADE, juntamente com todos os serviços necessários para hospedar e executar esses agentes. O conjunto de todos os containers, por sua vez, é chamado de plataforma. Trata-se de uma camada homogênea que oculta completamente dos agentes (a partir da aplicação) a complexidade e a diversidade das entidades subjacentes, tais como: hardware, sistema operacional, tipos de rede, JVM, dentre outros (TEIXEIRA, 2010).

O sistema de agentes do JADE é baseado no paradigma *peer-to-peer*. Cada agente é identificado por um nome globalmente único, definido como AID (*Agent Identifier*). O JADE gera esse identificador, concatenando um apelido definido pelo próprio usuário ao nome da plataforma, basicamente: <localname>@<hostname>:<port>/JADE (ex.apelido@G6C15:1099/JADE). Esse identificador pode ser associado ou desassociado a qualquer momento, e permite descobrir dinamicamente outros agentes a partir dos serviços de páginas brancas e de páginas amarelas, providos por agentes especializados da plataforma, conhecidos como AMS (*Agent Management System*) e DF (*Directory Facilitator*), respectivamente (TEIXEIRA, 2010).

2.4.3 Fundação para Agentes Físicos Inteligentes (FIPA)

A FIPA é uma associação internacional de companhias e organizações que compartilham esforços a fim de produzir especificações para tecnologias de agentes de software genéricas. Ela promove um conjunto de tecnologias para diferentes áreas de aplicação de tal forma que os desenvolvedores possam integrá-las para criar sistemas complexos com um alto grau de interoperabilidade (TEIXEIRA, 2010).

Para promover interoperabilidade entre diferentes plataformas, o JADE implementa todos os Protocolos de Transporte de Mensagens padrões (em inglês, *Message(s) Transport Protocol* - MTPs) definidos pela FIPA. Cada padrão inclui a definição de um

protocolo de transporte e uma codificação padronizada para o encapsulamento da mensagem (TEIXEIRA, 2010).

Quando a plataforma é inicializada, os quatro agentes especiais são automaticamente instanciados e inicializados pelo JADE. Seus papeis, definidos pelo padrão de Gerenciamento de Agente da FIPA, são descritos a seguir (SILVA, 2002):

- O *Agent Management System* (AMS) é o agente que supervisiona a plataforma inteira. Ele é o ponto de contato para todos os agentes que precisam interagir, a fim de acessar as páginas brancas da plataforma e gerenciar seu ciclo de vida. Todo agente criado é automaticamente registrado nas páginas brancas, gerenciadas pelo AMS, recebendo um AID válido;

- O *Directory Facilitator* (DF) é o agente que implementa o serviço de páginas amarelas, o qual é usado por qualquer agente que deseja registrar seus serviços ou buscar por outros disponíveis. O DF também aceita subscrições de agentes que desejam ser notificados quando um registro de serviço ocorre ou uma modificação for feita. DFs múltiplos podem ser inicializados concorrentemente a fim de distribuir o serviço de páginas amarelas através de vários domínios;

- O *Agent Communicattion Channel* (ACC) é o agente capaz de prover o caminho para que haja o contato básico entre agentes dentro e fora da plataforma. Ele provê o método de comunicação *default* do sistema que oferece um serviço de roteamento de mensagens confiável, ordenado e preciso, e

- O *Remote Monitoring Agent* (RMA) é o agente que monitora todas as ferramentas gráficas providas pela plataforma JADE, e que auxiliam os desenvolvedores, dentre outras contribuições, no gerenciamento dos agentes, acompanhando-os ao longo do ciclo de vida.

Detalhes de cunho mais operacional da plataforma JADE serão considerados no Capítulo 3 - REFERENCIAL TECNOLÓGICO.

2.5 Considerações Finais do Capítulo

Neste capítulo, foram apresentados os conceitos utilizados durante o processo de desenvolvimento desse trabalho. As definições de Sistemas Multiagentes e Aprendizado de Máquina objetivam apresentar uma visão geral sobre entidades de software inteligentes, que lidam com processamento distribuído, e podem ser implementadas usando recursos da IA, tal como algoritmos de aprendizagem e técnicas de aprendizado profundo. O processo de tomada de decisões por sistemas multiagentes exige que os agentes recebam treinamento

apropriado para o problema em questão. Este treinamento pode fazer uso de aprendizado de máquina aplicado à base de dados para alcançar um melhor resultado, e consequente mente, se aproximar da melhor decisão.

Dados os vários algoritmos disponíveis relacionados à implementação das Redes Neurais, cabe entender como funcionam os mais conhecidos e utilizados. Também é importante definir qual deles será utilizado no trabalho e qual será a sua finalidade.

No que se refere a sistemas multiagentes, é essencial entender como sua arquitetura funciona, bem como se dá a definição do comportamento de cada agente do sistema, e como esse agente pode se beneficiar de técnicas de aprendizado profundo, tais como Redes Neurais Artificiais e Raciocínio Baseado em Casos.

3 Referencial Tecnológico

Para o desenvolvimento deste projeto, foi necessária a utilização de ferramentas que auxiliaram no processo de desenvolvimento do software proposto, conforme consta em Ferramentas de Desenvolvimento de Software. Além disso, houve a necessidade de ferramentas adicionais para viabilizar a organização das atividades, comunicação e escrita da monografia, conforme colocado em Ferramentas de Apoio Geral. Por fim, têm-se as Considerações Finais do Capítulo.

3.1 Ferramentas de Desenvolvimento de Software

Muitos esforços da comunidade de engenharia de software têm sido dispendidos no intuito de fornecer apoio automatizado para as áreas de processo de desenvolvimento de software. Ferramentas, sistemas, e ambientes resultantes de tais esforços têm como objetivo auxiliar e/ou automatizar parte do processo de desenvolvimento de software para que o processo se torne mais predizível, menos dispendioso, mais fácil de gerenciar, produzindo produtos de maior qualidade (BROWN, 1992).

3.1.1 *Git e GitHub*

Um Sistema de Controle de Versão (VCS) monitora o histórico de alterações à medida que as pessoas e equipes colaboram em projetos em conjunto. Como os desenvolvedores fazem alterações no projeto, qualquer versão anterior do projeto pode ser recuperada a qualquer momento. Os VCSs fornecem a cada colaborador uma visão unificada e consistente de um projeto, evidenciando o trabalho que já está em andamento. Ver um histórico transparente das alterações, quem as fez, e como eles contribuem para o desenvolvimento de um projeto ajuda os integrantes da equipe manterem-se alinhados enquanto trabalham de forma independente (GitHub, Acesso em: 15 jun. 2024).

Em um sistema de controle de versão distribuído, cada desenvolvedor tem uma cópia completa do projeto e do histórico do projeto. Ao contrário dos sistemas de controle de versão centralizados conhecidos, os DVCSs não precisam de uma conexão constante com um repositório central. *Git* é o sistema de controle de versão distribuída mais popular. O *Git* é comumente usado para o desenvolvimento de software de código aberto e comercial, com benefícios significativos para indivíduos, equipes e negócios (GitHub, Acesso em: 15 jun. 2024).

Já o *GitHub* hospeda repositórios do *Git* e fornece aos desenvolvedores ferramentas para enviar um código melhor por meio das funcionalidades de linha de comando, proble-

mas(discussões encadeadas), *pull requests*, revisão de código ou o uso de uma coleção de aplicativos grátis, além de compras no *GitHub Marketplace*. Com camadas de colaboração como o fluxo de GitHub, uma comunidade de 100 milhões de desenvolvedores, e um ecossistema com centenas de integrações, *GitHub* muda a forma como o software é construído (GitHub, Acesso em: 15 jun. 2024).

GitHub cria colaboração diretamente no processo de desenvolvimento. O trabalho é organizado em repositórios onde os desenvolvedores podem definir os requisitos ou direção, bem como expectativas para os integrantes da equipe. Em seguida, ao usar o fluxo *GitHub*, os desenvolvedores simplesmente criam uma *branch* para trabalhar nas atualizações; enviar alterações para salvá-las; abrir um *pull request* para propor e discutir alterações, e fazer *merge* de *pull requests* quando todos estiverem na mesma página (GitHub, Acesso em: 15 jun. 2024).

3.1.2 *IntelliJ IDEA Community Edition*

IntelliJ IDEA é uma IDE voltada ao desenvolvimento de aplicações em diversas linguagens, entre elas, o Java. Trata-se de um ambiente multiplataforma que fornece experiência consistente nos sistemas operacionais *Windows*, *macOS* e *Linux* (JetBrains, Acesso em: 15 jun. 2024).

O *IntelliJ IDEA* está disponível nas seguintes edições, de acordo com (JetBrains, Acesso em: 15 jun. 2024):

- *Community Edition* é gratuito e de código aberto, licenciado sob Apache 2.0. Ele fornece todos os recursos básicos para desenvolvimento de aplicações java com a JVM e Android, e
- O *IntelliJ IDEA Ultimate* é comercial, distribuído com um período experimental de 30 dias. Ele fornece ferramentas e recursos adicionais para desenvolvimento web e empresarial.

Para os propósitos deste trabalho, foi utilizada a versão Community, pois ela já oferece todo o suporte para o desenvolvimento e a execução de aplicações *java*.

3.1.3 *Java, JDK e JVM*

Java é uma linguagem de programação orientada a objetos e uma plataforma de software amplamente utilizada que roda em bilhões de dispositivos, incluindo *notebooks*, dispositivos móveis, consoles de jogos, dispositivos médicos e muitos outros. As regras e a sintaxe do Java são baseadas nas linguagens C e C++ (IBM, Acesso em: 15 jun. 2024a).

O *Java Development Kit* (JDK) é um software para desenvolvedores Java. Inclui o interpretador Java, classes Java, e ferramentas de desenvolvimento Java, tais como: compilador, depurador, desmontador, visualizador de miniaplicativos, gerador de arquivo *stub* e gerador de documentação (IBM, Acesso em: 15 jun. 2024b).

O JDK permite escrever aplicativos que são desenvolvidos uma vez e executados em qualquer lugar, em qualquer máquina virtual Java. Os aplicativos Java desenvolvidos com o JDK em um sistema podem ser usados em outro sistema sem alterar ou recompilar o código. Os arquivos de classe Java são portáveis para qualquer máquina virtual Java padrão (IBM, Acesso em: 15 jun. 2024b).

A JVM é um ambiente de tempo de execução que pode ser incluído em um navegador da web ou em qualquer sistema operacional, como o IBM® i . A máquina virtual Java executa instruções geradas por um compilador Java. Consiste em um interpretador de *bytecode* e tempo de execução que permite que arquivos de classe Java sejam executados em qualquer plataforma, independentemente da plataforma em que foram originalmente desenvolvidos (IBM, Acesso em 15 jun. 2024).

O carregador de classes e o gerenciador de segurança, que atuam em tempo de execução, isolam o código que vem de outra plataforma. Eles também podem restringir quais recursos do sistema podem ser acessados pelas classes carregadas (IBM, Acesso em 15 jun. 2024).

3.1.4 *Apache Maven*

Maven é uma palavra iídiche, que significa acumulador de conhecimento. Esse ferramental começou como uma tentativa de simplificar os processos de construção no projeto *Jakarta Turbine*. Havia vários projetos, cada um com seus próprios arquivos de construção *Ant*, todos ligeiramente diferentes. Era necessário um padrão para se construir os projetos, ou seja, uma definição clara do que consistia o projeto, evidenciando uma maneira fácil de publicar informações do projeto e de compartilhar *Java Archives* (JARs), arquivo com as classes compiladas do *java*, entre vários projetos (Apache Maven, Acesso em: 15 jun. 2024).

O *Maven* constrói um projeto usando seu modelo de objeto de projeto (POM) e um conjunto de *plugins*. O *Maven* visa reunir os princípios atuais para o desenvolvimento de melhores práticas, e facilitar a orientação de um projeto nessa direção. Por exemplo, especificação, execução e relatório de testes unitários fazem parte do ciclo normal de construção usando *Maven* (Apache Maven, Acesso em: 15 jun. 2024).

3.1.5 *JAVA Agent DEvelopment Framework*

Java Agent DEvelopment Framework (JADE) é um *Framework* de *software* totalmente implementado na linguagem *Java*. Simplifica a implementação de sistemas multi-agentes através de um *middleware* que atende às especificações FIPA, e através de um conjunto de ferramentas gráficas que auxiliam nas fases de depuração e implantação. Um sistema baseado em JADE pode ser distribuído entre máquinas (que nem precisam compartilhar o mesmo sistema operacional), e a configuração pode ser controlada através de uma Interface Gráfica do Utilizador (GUI) remota . A configuração pode até ser alterada em tempo de execução, movendo os agentes de uma máquina para outra, como e quando necessário. JADE é totalmente implementado em linguagem *Java* e o requisito mínimo do sistema é a versão 5 do *JAVA* (o ambiente de tempo de execução ou JDK) (Telecom Italia Lab, Acesso em: 15 jun. 2024).

Além da abstração do agente, o JADE fornece um modelo de composição e execução de tarefas simples; comunicação de agente ponto a ponto baseada no paradigma de passagem de mensagens assíncrona; serviço de páginas amarelas que possui mecanismo de descoberta de publicação e assinatura, dentre outros recursos avançados que facilitam o desenvolvimento de um sistema distribuído (Telecom Italia Lab, Acesso em: 15 jun. 2024).

3.1.5.1 Agentes

A noção fraca de agente denota um sistema de computador baseado em software com as seguintes propriedades, segundo Wooldridge e Jennings (1995):

- Autonomia: os agentes operam sem a intervenção direta de humanos ou outros, e têm algum tipo de controle sobre suas ações e estado interno;
- Habilidade social: os agentes interagem com outros agentes (e possivelmente humanos) por meio de alguma linguagem de comunicação de agente;
- Reatividade: os agentes percebem seu ambiente e respondem de forma oportuna às mudanças que ocorrem nele, e
- Proatividade: além de agir em resposta ao seu ambiente, os agentes são capazes de exibir comportamento direcionado a objetivos, tomando a iniciativa.

A noção forte de agente é uma extensão da noção mais fraca, e defende propriedades humanísticas e mentais adicionais, como crença, desejo e intenção (SHOHAM, 1993).

Os agentes residem em uma plataforma que, em consonância com a visão apresentada, fornece aos agentes um mecanismo adequado para se comunicarem por meio de

nomes, independentemente da complexidade e da natureza do ambiente subjacente (ou seja, sistemas operacionais, redes, dentre outros) (NIKRAZ; CAIRE; BAHRI, 2006).

3.1.5.2 Descobridor de Agentes

A descoberta de agentes pode ser realizada por meio de convenções de nomenclatura apropriadas. Porém, uma maneira mais sofisticada de resolver o problema da descoberta de agentes é a adoção de um mecanismo de "páginas amarelas". Isso permite a descoberta de agentes com base em suas características, por exemplo, os serviços que eles fornecem. Um mecanismo de páginas amarelas pode ser totalmente distribuído entre todos os agentes no sistema ou centralizado com um único agente (com um nome conhecido) responsável por ele. Mesmo que essa escolha, neste ponto, seja uma escolha de *design* de alto nível, considerando que a metodologia proposta visa a plataforma JADE, é altamente recomendável adotar uma abordagem centralizada. Esta abordagem mapeia completamente para o agente facilitador de diretório fornecido pelo JADE e, assim, economiza muito trabalho nas fases sucessivas do processo de desenvolvimento (NIKRAZ; CAIRE; BAHRI, 2006).

3.1.5.3 Ontologia

Quando os agentes no sistema interagem, eles trocam informações que se referem a entidades, abstratas ou concretas, que existem no ambiente onde os agentes residem. Essas entidades podem ser primitivas, como uma *String* ou um número; ou podem ter estruturas complexas definidas por modelos especificados em termos de um nome e um conjunto de *slots*. Os valores desses *slots* devem ser de um determinado tipo. Esses modelos de entidades complexas são referidos como Conceitos (NIKRAZ; CAIRE; BAHRI, 2006).

Além disso, as entidades geralmente estão relacionadas por meio de relações que podem ser verdadeiras ou falsas. Semelhante a entidades complexas, as relações também têm estruturas definidas por modelos e, novamente, esses modelos são especificados em termos de um nome e um conjunto de *slots*, cujos valores devem ser de um determinado tipo. Esses modelos de relação são referidos como Predicados, e devem considerar o estudo de caso em questão. Finalmente, um tipo particular de entidade complexa é representado por descritores de ações que os agentes podem executar (NIKRAZ; CAIRE; BAHRI, 2006).

Ações, quando executadas, podem produzir um efeito e/ou gerar um resultado para ser enviado de volta ao solicitante. Uma ontologia é um conjunto de conceitos, predicados e ações de agentes referentes a um determinado domínio (NIKRAZ; CAIRE; BAHRI, 2006).

3.1.6 *Apache OpenNPL*

A biblioteca Apache OpenNLP é um conjunto de ferramentas baseado em aprendizado de máquina para o processamento de texto em linguagem natural. Ele oferece suporte às tarefas mais comuns de Processamento de Linguagem Natural (PLN), como *tokenização*, segmentação de frases, marcação de classe gramatical, extração de entidade nomeada, fragmentação, análise e resolução de correferência. Essas tarefas geralmente são necessárias para construir serviços de processamento de texto mais avançados. O *OpenNLP* também inclui entropia máxima[1] e aprendizado de máquina baseado em *perceptron*[2] (Apache Software Foundation, 2024).

A biblioteca *Apache OpenNLP* contém vários componentes, permitindo construir um *pipeline* completo de processamento de linguagem natural. Esses componentes incluem: detector de frases, *tokenizador*, localizador de nomes, categorizador de documentos, etiquetador de classe gramatical, *chunker*, analisador, resolução de correferência. Os componentes contêm partes que permitem executar a respectiva tarefa de processamento de linguagem natural, treinar um modelo e, muitas vezes, também avaliar um modelo. Cada um desses recursos é acessível por meio de sua interface de programa de aplicativo (API). Além disso, uma Interface de Linha de Comando (CLI) é fornecida para conveniência de experimentos e treinamento (Apache Software Foundation, 2024).

Como no projeto proposto, a base de dados está escrita em português brasileiro (PT-BR), faz-se necessária a utilização de ferramentas que forneçam suporte e que se adéquem às particularidades da língua portuguesa, como as acentuações.

3.1.6.1 Detector de Idioma

O *OpenNLP Language Detector* classifica um documento em idiomas ISO-639-3 de acordo com os recursos do modelo. Um modelo pode ser treinado com algoritmos *Maxent*, *Perceptron* ou *Naive Bayes*. Por padrão, normaliza um texto e o gerador de contexto extrai n-gramas de tamanho 1, 2 e 3. Os tamanhos de n-gramas, a normalização e o gerador de contexto podem ser customizados estendendo o *LanguageDetectorFactory* (Apache Software Foundation, 2024).

Os normalizadores padrão estão apresentados no Quadro 1. Por exemplo, tem-se normalizador para substituição de emojis por espaço em branco, chamado *EmojiCharSequenceNormalizer*, e normalizador que reduz caracteres que se repetem três ou mais vezes para apenas duas repetições, chamado *ShrinkCharSequenceNormalizer*.

[1] É um princípio em aprendizado de máquina que visa maximizar a incerteza nas previsões de um modelo, permitindo que ele mantenha a maior variabilidade possível (Apache Software Foundation, 2024).

[2] É um modelo básico de rede neural que simula um neurônio biológico. Ele consiste em múltiplas entradas, cada uma associada a um peso, e produz uma saída binária usando uma função de ativação.(ROSENBLATT, 1958)

Quadro 1 – Normalizadores

Normalizador	Descrição
EmojiCharSequenceNormalizer	Substitui *emojis* por espaço em branco.
UrlCharSequenceNormalizer	Substitui URLs e *e-mails* por um espaço em branco.
TwitterCharSequenceNormalizer	Substitui *hashtags* e nomes de usuário do Twitter por espaços em branco.
NumberCharSequenceNormalizador	Substitui sequências numéricas por espaços em branco.
ShrinkCharSequenceNormalizer	Reduza os caracteres que se repetem três ou mais vezes para apenas duas repetições.

Fonte: Apache Software Foundation (2024) - Tradução (Autor)

3.1.6.2 Detecção de Frases

O *OpenNLP Sentence Detector* pode detectar se um caractere de pontuação marca o final de uma frase ou não. Nesse sentido, uma frase é definida como a sequência de caracteres mais longa com espaço em branco entre dois sinais de pontuação. A primeira e a última frase abrem uma exceção a esta regra. O primeiro caractere que não seja um espaço em branco é considerado o início de uma frase, e o último caractere que não seja um espaço em branco é considerado o final de uma frase (Apache Software Foundation, 2024).

3.1.6.3 API de Treinamento

O *Sentence Detector* também oferece uma API para treinar um novo modelo de detecção de frases. Basicamente, são necessários três passos para treiná-lo:

1. O aplicativo deve abrir um fluxo de dados de amostra;
2. O aplicativo deve chamar o método *SentenceDetectorME.train*, e
3. O aplicativo deve salvar o *SentenceModel* em um arquivo ou usá-lo diretamente.

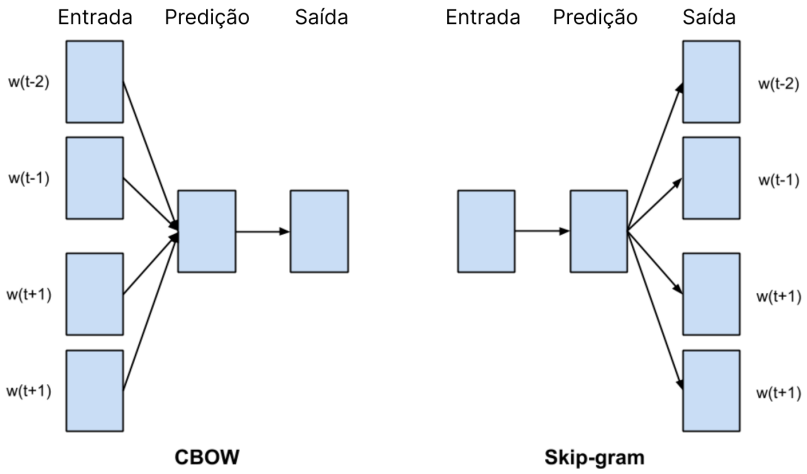
3.1.7 *Word2Vec*

Word2vec é uma rede neural de duas camadas que processa texto. Sua entrada é um corpus de texto, e sua saída é um conjunto de vetores: vetores de características para palavras nesse corpus. Embora o *Word2vec* não seja uma rede neural profunda, ele transforma o texto em uma forma numérica que as redes profundas podem entender. *Deeplearning4j* implementa uma forma distribuída de *Word2vec* para Java e Scala, que funciona em Spark com GPUs (Konduit AI, 2024).

Com dados, uso e contextos suficientes, o *Word2vec* pode fazer suposições altamente precisas sobre o significado de uma palavra com base em aparências anteriores. Essas suposições podem ser usadas para estabelecer a associação de uma palavra com outras palavras (por exemplo, “homem” está para “menino” o que “mulher” está para “menina”), ou agrupar documentos e classificá-los por tópico. Esses *clusters* podem formar a base de pesquisas, análises de sentimentos e recomendações em campos tão diversos como pesquisa científica, descoberta jurídica, comércio eletrônico e gestão de relacionamento com clientes (Konduit AI, 2024).

Cabe colocar que o treinamento pode ocorrer de duas maneiras: usando o contexto para prever uma palavra-alvo - método conhecido como “sacola contínua” de palavras (CBOW); ou usando uma palavra para prever um contexto-alvo, que é chamado de *skip-gram*. Ambos os métodos são apresentados na Figura 21. Utilizou-se, no presente trabalho, o último método, pois o mesmo tende a produzir resultados mais precisos em grandes conjuntos de dados (Konduit AI, 2024).

Figura 2 – CBOW e Skip-gram



Fonte: Konduit AI (2024) - Tradução (Autor)

O *Word2Vec* foi utilizado para melhorar a análise do texto em português, auxiliando no processo de representação do texto e no desempenho das tarefas de classificação de sentimentos.

3.1.7.1 Anatomia do *Word2vec* em DL4J

Dos componentes de processamento de linguagem natural do *Deeplearning4j*, têm-se, segundo Konduit AI (2024):

- *SentenceIterator/DocumentIterator*: usado para iterar em um conjunto de dados. Um *SentenceIterator* retorna *strings*, e um *DocumentIterator* funciona com fluxos

de entrada;

- *Tokenizer/TokenizerFactory*: usado na tokenização do texto. Nos termos do PLN, uma frase é representada como uma série de *tokens*. Um *TokenizerFactory* cria uma instância de um *tokenizer* para uma "frase";
- *VocabCache*: usado para rastrear metadados, incluindo contagens de palavras, ocorrências de documentos, conjunto de *tokens* (não vocabulário neste caso, mas sim *tokens* que ocorreram), vocabulário (recursos incluídos no pacote de palavras, bem como na tabela de pesquisa de vetor de palavras), e
- Índice invertido: armazena metadados sobre onde as palavras ocorreram. Pode ser usado para compreender o conjunto de dados.

Embora *Word2vec* refira-se a uma família de algoritmos relacionados, esta implementação usa Amostragem Negativa. Nessa amostragem, ocorre a seleção de exemplos negativos, ou seja, palavras que não estão relacionadas ao contexto em questão, a fim de treinar o modelo de forma mais eficiente.

3.1.7.2 Tokenizando os Dados

Word2vec precisa ser alimentado com palavras em vez de frases inteiras. Sendo assim, a próxima etapa é tokenizar os dados. Tokenizar um texto é dividi-lo em suas unidades atômicas, criando um novo *token* cada vez que se atinge, por exemplo, um espaço em branco (Konduit AI, 2024).

3.1.7.3 Treinando o Modelo

Com os dados prontos, é possível configurar a rede neural *Word2vec* e alimentar os tokens. A configuração aceita vários hiperparâmetros. Alguns requerem alguma explicação, segundo Konduit AI (2024):

- *batchSize* é a quantidade de palavras processada por vez;
- *minWordFrequency* é o número mínimo de vezes que uma palavra deve aparecer no corpus. Aqui, se aparecer menos de 5 vezes, não se aprende. As palavras devem aparecer em vários contextos para aprender recursos úteis sobre elas;
- *useAdaGrad - Adagrad* cria um gradiente diferente para cada recurso;
- *layerSize* especifica o número de recursos no vetor de palavras. Isso é igual ao número de dimensões no *featurespace*. Palavras representadas por 500 características tornam-se pontos em um espaço de 500 dimensões;

- *learningRate* é o tamanho do passo para cada atualização dos coeficientes, à medida que as palavras são reposicionadas no espaço de recursos;
- *minLearningRate* é o piso da taxa de aprendizagem. A taxa de aprendizagem diminui à medida que o número de palavras que você treina diminui. Se a taxa de aprendizagem diminuir muito, a aprendizagem da rede não será mais eficiente. Isso mantém os coeficientes em movimento;
- *iterate* informa à rede em qual lote do conjunto de dados ele está treinando;
- *tokenizer* alimenta as palavras do lote atual, e
- *vec.fit()* informa à rede configurada para iniciar o treinamento.

3.1.8 *Deeplearning4j*

Eclipse Deeplearning4j (DL4J) é um conjunto de ferramentas para executar aprendizado profundo na JVM. É a única estrutura que permite treinar modelos de java, enquanto interopera com o ecossistema *python*. Essa interoperabilidade dá-se por meio de uma combinação de execução *python* e ligações *cpython*. Adicionalmente, têm-se suporte à importação de modelo e interoperabilidade com outros suportes, tais como: *TensorFlow Java* e *ONNIX Runtime* (Konduit AI, 2024).

TensorFlow Java[3] é uma biblioteca de código aberto para desenvolvimento, treinamento e implantação de modelos de aprendizado de máquina. Dispõe de uma API para auxiliar quem desejar usá-la.

ONNX Runtime[4] é um acelerador de modelos de aprendizado de máquina, que possui uma interface flexível para integração com diferentes tipos de hardware.

O *Deeplearning4j* é uma linguagem de domínio específico para configurar redes neurais profundas, que são feitas de múltiplas camadas. Tudo começa com um *MultiLayerConfiguration*, que organiza essas camadas e seus hiperparâmetros. Hiperparâmetros são variáveis que determinam como uma rede neural aprende. Eles incluem quantas vezes atualizar os pesos do modelo; como inicializar esses pesos; qual função de ativação anexar aos nós; qual algoritmo de otimização usar, e quão rápido o modelo deve aprender (KONDUIT AI, 2024).

Deeplearning4j possui vários submódulos, segundo Konduit AI (2024), incluindo:

- *Samediff*: uma estrutura semelhante a *tensorflow/pytorch* para execução de gráficos complexos. Esta estrutura é de nível inferior, mas muito flexível. Representa ainda a API base para executar gráficos *onnx* e *tensorflow*;

[3] Disponível em: <https://www.tensorflow.org/install/lang_java_legacy>. Acesso em: 3 jun 2024.
[4] Disponível em: <https://onnxruntime.ai>. Acesso em: 3 jun 2024.

- *Nd4j*: *numpy++* para java. Contém uma combinação de operações *numpy* e operações *tensorflow/pytorch*;

- *Libnd4j*: uma biblioteca C++ leve e independente que permite que o código matemático seja executado em diferentes dispositivos. Otimizável para execução em uma ampla variedade de dispositivos;

- *Python4j*: uma estrutura de execução de *script python* que facilita a implantação de *scripts python* na produção;

- Integração *Apache Spark*: uma integração com a estrutura *Apache Spark* que permite a execução de *pipelines* de aprendizado profundo no *Spark*, e

- *Datavec*: uma biblioteca de transformação de dados que converte dados brutos de entrada em tensores adequados para execução de redes neurais.

3.1.8.1 *Doc2Vec*

O principal objetivo do *Doc2Vec* é associar documentos arbitrários a rótulos. Portanto, rótulos são obrigatórios. *Doc2vec* é uma extensão do *word2vec* que aprende a correlacionar rótulos e palavras, em vez de palavras com outras palavras. A implementação do *Deeplearning4j* visa atender as comunidades *Java*, *Scala* e *Clojure* (KONDUIT AI, 2024).

O primeiro passo é criar um vetor que represente o "significado" de um documento, que pode então ser usado como entrada para um algoritmo de aprendizado de máquina supervisionado para associar documentos a rótulos. No padrão de construtor *ParagraphVectors*, o *labels()* é o método que aponta para os rótulos nos quais treinar (KONDUIT AI, 2024).

3.1.8.2 *Sentence Iterator*

Trata-se de um iterador que representa pedaços de texto em uma rede neural na forma de vetores e também abrange o conceito de documentos no processamento de texto. No processamento de linguagem natural, um documento ou frase é normalmente usado para encapsular um contexto que um algoritmo deve aprender (KONDUIT AI, 2024).

Dependendo de como a entrada é processada, a saída de um iterador de frase será então passada para um *tokenizer* para o processamento de *tokens* individuais, que geralmente são palavras, mas também podem ser *ngrams*, *skipgrams* ou outras unidades. O *tokenizer* é criado por frase por uma fábrica de *tokenizers*. A fábrica do *tokenizer* é o que é passado para um vetorizador de processamento de texto (KONDUIT AI, 2024).

3.1.8.3 *Tokenization*

Tokenização é o processo de dividir o texto em palavras individuais. As janelas do Word também são compostas por *tokens*. *Word2Vec* pode gerar janelas de texto que incluem exemplos de treinamento para entrada em redes neurais (KONDUIT AI, 2024).

3.1.8.4 *Vocabulary Cache*

O *cache* de vocabulário, ou cache de vocabulário, é um mecanismo para lidar com tarefas de linguagem natural de uso geral no *Deeplearning4j*, incluindo o Termo Frequência-Frequência Inversa do Documento TF-IDF[5] normal, vetores de palavras e certas técnicas de recuperação de informações. O objetivo do cache de vocabulário é ser um local único para vetorização de texto, encapsulando técnicas comuns a conjuntos de palavras e vetores de palavras, entre outros (KONDUIT AI, 2024).

O *cache Vocab* lida com o armazenamento de *tokens*, frequências de contagem de palavras, frequências de documentos inversos e ocorrências de documentos por meio de um índice invertido. O *InMemoryLookupCache* é a implementação de referência (KONDUIT AI, 2024).

3.1.9 *Makefile*

Makefiles são arquivos que permitem organizar os comandos da compilação nos projetos de software, tais como a ligação e a montagem de arquivos dos projetos. Este tutorial é um pequeno guia para compilar pequenos e médios projetos (RAMOS, 2015).

Em um *Makefile*, a escrita do código é feita em blocos que possuem três elementos importantes, segundo Ramos (2015):

- alvo: arquivo a ser construído. Exemplos de alvos são arquivos-objeto ou executáveis;
- pré-requisito: arquivo necessário para gerar o alvo. Um alvo pode ter nenhum, e um ou mais pré-requisitos, e
- comando: ação aplicada ao(s) pré-requisito(s) para gerar o alvo.

3.1.10 *Docling*

O *Docling* analisa documentos e os converte para o formato desejado. Ele lê formatos de documentos populares, tais como *PDF*, *DOCX*, *PPTX*, *XLSX*, *JPG*, *HTML*, *AsciiDoc* e *Markdown* e exporta para *HTML*, *Markdown*, *JSON* ou texto puro (*TXT*) (DS4SD Team, 2025).

[5] Nesse método, a importância dos termos é proporcional a frequência de ocorrência dos mesmos em cada documento da coleção e inversamente proporcional ao número total de documentos em que os termos aparecem (BALDASSIN; GUILHERME; MALTEMPI, 2002).

3.1.11 *Astah Community*

Astah Community é um software para modelagem UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada) com suporte a UML 2, desenvolvido pela *Change Vision, Inc* e disponível para sistemas operacionais Windows 64 bits. Anteriormente conhecido por JUDE, um acrônimo de *Java and UML Developers Environment* (Ambiente para Desenvolvedores UML e Java). *Astah Community* disponibiliza para desenvolvimento, os diagramas de Classes, Casos de Uso, Sequência, Comunicação, Máquina de Estados, Atividade, Componentes, Implantação e Diagrama de Estrutura Composta (Techtudo, 2025).

3.1.12 *Visual Paradigm Online*

O *Visual Paradigm Online* é uma ferramenta baseada na nuvem para modelagem visual, *design* e colaboração. Ele é uma versão online do *Visual Paradigm*, um software amplamente utilizado para *design* de processos e modelagem de banco de dados. Nesse trabalho, ele foi utilizando para a modelagem de processos de negócio com BPMN (Visual Paradigm, 2025).

3.2 Ferramentas de Apoio Geral

Seguem algumas ferramentas adicionais, que colaboram no projeto para organização das atividades, bem como comunicação e escrita dessa monografia.

3.2.1 *Trello*

O *Trello* tem suas origens no trabalho transformador de melhoria contínua da *Toyota*, Taaichi Onoh, e demais empresas associadas. Eles criaram uma versão digital do sistema *kanban*, um processo de fluxo de trabalho baseado em cartões de notas para orientação e otimização. O *kanban* é um sistema simples. Entretanto, para usar essas ferramentas de forma mais eficaz, é útil ter um conhecimento básico dos princípios que orientam esse sistema (KAUR, 2018).

Os quadros *kanban* dividem todos os passos necessários para organizar um projeto em cartões - originalmente cartões de notas escritos pelos membros da equipe e passados junto com o inventário ou itens fabricados em cada etapa do processo de produção. Ele descreve cada item de trabalho específico, quem é responsável por completá-lo, seu cronograma e status atual, e qualquer outra informação relacionada e necessária. Cada cartão, em forma digital, é organizado em uma linha do tempo com base nas expectativas da equipe do projeto e nas dependências de cada tarefa dentro do projeto como um todo (KAUR, 2018).

Trello é uma ferramenta de colaboração e gerenciamento de projetos baseada na *web*, disponível em três versões: Grátis, *Business Class* e *Enterprise*. As versões *Business Class* e *Enterprise* do *Trello* oferecem um conjunto expandido de recursos. Para o desenvolvimento deste projeto, foram utilizados os recursos disponíveis na versão gratuita. O *Trello* é organizado como uma espécie de quadro de avisos virtual, onde é possível colocar avisos para tarefas específicas e marcos-chave para diferentes projetos, criando uma linha do tempo ou lista de tarefas, e então atualizar, gerenciar e mover os avisos conforme o projeto avança. Dentro do *Trello*, é possível criar vários quadros pessoais e em grupo para acompanhar itens e atualizações usando listas e cartões que podem ser movidos pela tela (KAUR, 2018).

3.2.2 *Microsoft Teams*

Microsoft Teams é uma das plataformas em que estudantes e professores podem integrar todo o seu trabalho em um só lugar. *Microsoft Teams* é uma plataforma de aprendizado que integra chats, informações, tarefas e aplicativos em um único local para que professores e funcionários possam acompanhar o progresso dos alunos (SITUMORANG, 2020).

Esta plataforma pode ser usada por instituições educacionais para qualquer pessoa que tenha ativado uma conta escolar ou universitária para aproveitar as capacidades disponíveis ao usar o *Microsoft Teams* como uma plataforma de aprendizado *online* para construir um sistema de gerenciamento de aprendizado acadêmico. Recursos similares estão disponíveis na forma de ambientes de ensino e aprendizado bem integrados, e o *Microsoft Teams* é uma adequada plataforma de aprendizado (FUADDAH; MAHARANI, 2021).

O *Microsoft Teams* auxilia os professores a otimizar o fluxo de aprendizado, resultando em ambientes de aprendizado mais eficazes. Nesse cenário, o aprendizado *online* tem um impacto positivo nas atitudes dos alunos na plataforma *Microsoft Teams* (FUADDAH; MAHARANI, 2021).

Diante do exposto, essa plataforma foi utilizada para comunicação semanal entre orientadores e orientando ao longo do trabalho como um todo.

3.2.3 *Slack*

Uma solução para os problemas de comunicação e colaboração é o *Slack*. Trata-se de um espaço de trabalho digital que promove a colaboração e a discussão de maneira unificada. Ele visa substituir as muitas plataformas de comunicação diferentes que existem no ambiente de trabalho (e na sala de aula): e-mail, mensagem de texto, mensageiro instantâneo, e assim por diante (TECKCHANDANI, 2018).

O uso do *Slack* no ambiente de trabalho está aumentando rapidamente. O *Slack* está sendo usado por mais de 50.000 empresas em todo o mundo, incluindo 43 das empresas na Fortune 100. Um espaço de trabalho do *Slack* pode ser acessado através de um navegador da web, um aplicativo de *desktop* (para *Windows*, *Mac* e *Linux*) ou um aplicativo para *smartphone* (para *iPhone*, *Android* e *Windows Phone*). Embora existam versões pagas do *Slack*, a versão gratuita é mais do que suficiente para ser usada como uma ferramenta de aprendizado baseada em sala de aula (TECKCHANDANI, 2018).

O *Slack* permite que os membros da equipe se comuniquem dentro do espaço de trabalho usando dois métodos. O primeiro é se comunicar por meio de canais. Canais são como salas de *chat*: todos que fazem parte do canal podem conversar entre si. Os membros podem criar seus próprios canais. Os canais podem ser públicos (acessíveis por todos os membros) ou privados (restritos a um conjunto específico de membros). Os canais privados são análogos a um e-mail, onde várias pessoas são incluídas como cópias. Mas, ao contrário do e-mail, é fácil adicionar e remover pessoas da conversa, e as pessoas que não fazem parte do canal privado não podem se adicionar a ele ou ver que o canal existe. Em um ambiente de sala de aula, isso significa que até mesmo o professor pode não ter acesso a todos os canais no espaço de trabalho (TECKCHANDANI, 2018).

O segundo método de comunicação é através de mensagens diretas. As mensagens diretas são para conversas que os membros desejam ter fora dos canais. Até oito pessoas podem ser enviadas uma mensagem ao mesmo tempo. Em vez de digitar, uma mensagem direta também pode ser uma chamada de voz ou vídeo. As mensagens diretas são mais úteis para conversas rápidas e privadas (TECKCHANDANI, 2018).

Para a desenvolvimento desse trabalho, foi sido utilizado majoritariamente o primeiro método, via canais públicos e privados. Os canais públicos tratam de informativos e espaço para a postagem de dúvidas e respostas. Já os canais privados são utilizados para a comunicação direta dos orientandos com os professores orientadores. Utilizou-se ainda do recurso de envio de mensagens diretas entre os integrantes do grupo.

3.2.4 *Telegram*

O *Telegram* é um aplicativo de mensagens com foco no envio instantâneo de mensagens, simples e gratuito. É possível utilizar o *Telegram* em vários dispositivos ao mesmo tempo — as mensagens são sincronizadas perfeitamente em qualquer quantidade de telefones, *tablets* ou computadores conectados na mesma conta. O *Telegram* tem mais de 700 milhões de usuários ativos mensais, sendo um dos dez aplicativos mais baixados do mundo (Telegram Messenger LLP, Acesso em: 15 jun. 2024).

Com o *Telegram*, é possível enviar mensagens, fotos, vídeos e arquivos de qualquer tipo (*doc*, *zip*, *mp3*, etc), além de criar grupos de até 200.000 pessoas ou canais para

transmitir para públicos ilimitados. É possível escrever para contatos da agenda e encontrar pessoas pelos nomes de usuário delas. Como resultado, o *Telegram* é como *SMS* e *e-mail* combinados. Além disso, é oferecido suporte a chamadas de voz e videochamadas criptografadas de ponta a ponta, bem como *chats* de voz em grupos para participantes (Telegram Messenger LLP, Acesso em: 15 jun. 2024).

Esses vários recursos do *Telegram* foram relevantes para facilitar a comunicação do autor com terceiros, sejam professores orientadores, sejam demais graduandos.

3.2.5 *Overleaf*

Overleaf é uma *startup* e empresa social que cria ferramentas modernas de autoria colaborativa para cientistas – como o *Google Docs for Science*. Seu principal produto é um editor colaborativo, também chamado de *Overleaf*, que funciona de modo *on-line* (em tempo real) para artigos, teses, relatórios técnicos e outros documentos escritos na linguagem de marcação LaTeX (OVERLEAF, 2024b). Foi utilizado na escrita dessa monografia.

3.2.5.1 LaTeX

O LaTeX (pronuncia-se “LAY -tek” ou “LAH -tek”) é uma ferramenta para composição tipográfica de documentos com aparência profissional. No entanto, o modo de operação do LaTeX é bastante diferente de muitos outros aplicativos de produção de documentos, como o *Microsoft Word* ou o *LibreOffice Writer*. Tais ferramentas fornecem aos usuários uma página interativa na qual eles digitam e editam seu texto e aplicam várias formas de estilo (OVERLEAF, 2024a).

Já no LaTeX, há um benefício importante que é a separação do conteúdo do documento do estilo do documento. Sendo assim, depois de ter escrito o conteúdo do seu documento, sua aparência pode ser alterada com facilidade. Da mesma forma, pode-se criar um arquivo LaTeX que defina o *layout*/estilo de um determinado tipo de documento. Esse arquivo pode ser usado como modelo para padronizar a autoria/produção de documentos adicionais desse tipo; por exemplo, isso permite que editores científicos criem modelos de artigos. Outros autores usam esses modelos para escrever artigos para submissão a periódicos (OVERLEAF, 2024a). Essa ferramenta foi relevante para a escrita da presente monografia.

3.3 Considerações Finais do Capítulo

Neste capítulo, foram apresentados os conceitos relacionados às tecnologias que auxiliaram no processo de comunicação e desenvolvimento do projeto, bem como as bibliotecas e os *frameworks* que fornecem as implementações que serão utilizadas pelo projeto.

Para o processo de desenvolvimento, buscou-se utilizar ferramentas *open source* e com suporte da comunidade. Dado que a base do projeto é desenvolvida em Java, buscou-se utilizar *frameworks* e bibliotecas já consolidadas, como o *Deeplearning4j*, que oferece um conjunto de funcionalidades de *deeplearning* para Processamento de Linguagem Natural, e modelos já treinados em português do *Long Short-Term Memory*.

No Quadro 2, segue um resumo das principais ferramentas e suas respectivas versões e descrições.

Quadro 2 – Principais ferramentas utilizadas no trabalho

Ferramenta	**Versão**	**Descrição/Propósito**
Git	2.34.1	Versionamento do código-fonte
GitHub	3.12.4	Hospedagem e gerenciamento do código-fonte
IntelliJ IDEA Community Edition	2023.3.2	Ambiente de desenvolvimento integrado (IDE), projetado para desenvolver aplicativos em Java
Java	21.0.2	Linguagem de programação orientada a objetos
JDK	21.0.2+13-LTS-58	Conjunto de ferramentas de desenvolvimento para aplicativos Java
JVM	21.0.2+13-LTS-58	Converte o *bytecode* Java em instruções de máquina específicas do sistema operacional subjacente
Apache Maven	3.9.2	Realiza a automação, gerenciamento e a gestão de dependências de projetos Java usando um arquivo de configuração *pom.xml*
JADE	4.3	É uma plataforma de desenvolvimento para criar sistemas multiagentes em Java
Apache OpenNPL	2.3.3	É uma biblioteca de código aberto para processamento de linguagem natural
Word2Vec	1.0.0-beta7	É um algoritmo para aprendizado de representações vetoriais de palavras a partir de grandes quantidades de texto
Deeplearning4j	1.0.0-M2.1	É uma biblioteca de código aberto para aprendizado profundo desenvolvida em Java
Makefile	0.77	Automatiza o processo de compilação e construção de programas e aplicativos
Docling	2.14.0	Converte PDF para texto puro (TXT)
Astah Community	7.2.0	Modelagem UML
Visual Paradigm Online	17.2	Modelagem de processos BPMN
Trello	2024.10.4.22478	É uma plataforma de gerenciamento de projetos baseada em quadros
Microsoft Teams	Maio, 2024	É uma biblioteca de código aberto para processamento de linguagem natural
Slack	4.38.125	É uma plataforma que permite a colaboração em equipe e a comunicação interna em empresas e organizações
Telegram	v10.0	É um aplicativo de mensagens instantâneas, voz sobre IP e serviço de rede social baseado na nuvem
Overleaf	4.2.4	É uma plataforma de edição de documentos LaTeX baseada na web
LaTeX	2023/11/01	É uma linguagem de marcação utilizada para a preparação de documentos técnicos e científicos

Fonte: Próprio Autor

4 Metodologia

Este capítulo descreve a metodologia utilizada na realização desta pesquisa. De início, é apresentada a Classificação da Pesquisa, que detalha os conceitos da pesquisa relacionados a abordagem, natureza, objetivos e procedimentos. Depois, segue-se para o Método Investigativo, que apresenta informações relacionadas à literatura consultada durante a concepção deste trabalho. Seguindo, o Método de Desenvolvimento evidencia como ocorreu o processo de desenvolvimento, e a Método de Análise de Resultados apresenta como os resultados foram obtidos e avaliados. Por fim, será apresentado um detalhamento sobre o Cronograma das Atividades e as Considerações Finais do Capítulo.

4.1 Classificação da Pesquisa

Para Fonseca (2002), *methodos* significa organização, e *logos*, estudo sistemático, pesquisa, investigação; ou seja, metodologia é o estudo da organização, dos caminhos a serem percorridos, para se realizar uma pesquisa ou um estudo, ou para se fazer ciência. Etimologicamente, significa o estudo dos caminhos, dos instrumentos utilizados para fazer uma pesquisa científica.

A pesquisa é a atividade nuclear da Ciência. Ela possibilita uma aproximação e um entendimento da realidade a investigar. Ela é o resultado de um inquérito ou exame minucioso, realizado com o objetivo de resolver um problema, recorrendo a procedimentos científicos (GERHARDT; SILVEIRA, 2009).

Ainda segundo Gerhardt e Silveira (2009), as pesquisas científicas podem ser classificadas quanto à sua abordagem, sua natureza, seus objetivos e seus procedimentos.

4.1.1 Abordagem

Em relação à abordagem, as pesquisas científicas podem classificadas em Qualitativa e Quantitativa, segundo Gerhardt e Silveira (2009):

- Qualitativa: A pesquisa qualitativa não se preocupa com representatividade numérica, mas, sim, com o aprofundamento da compreensão de um grupo social, de uma organização, entre outros. Os pesquisadores que adotam a abordagem qualitativa opõem-se ao pressuposto que defende um modelo único de pesquisa para todas as ciências, já que as ciências sociais têm sua especificidade, o que pressupõe uma metodologia própria. Assim, os pesquisadores qualitativos recusam o modelo positivista aplicado ao estudo da vida social, uma vez que o pesquisador não pode fazer

julgamentos nem permitir que seus preconceitos e crenças contaminem a pesquisa (GOLDENBERG, 1997), e

- Quantitativa: Diferentemente da pesquisa qualitativa, os resultados da pesquisa quantitativa podem ser quantificados. Como as amostras geralmente são grandes e consideradas representativas da população, os resultados são tomados como se constituíssem um retrato real de toda a população alvo da pesquisa. A pesquisa quantitativa centra-se na objetividade (FONSECA, 2002).

Nesse trabalho, foi realizada uma pesquisa, predominantemente, Quantitativa, uma vez que a pesquisa envolve a análise de uma base de dados estruturada. Também foi empregando a utilização de aprendizado de máquina e aprendizado profundo, que utiliza essas bases de dados para treinamento, aplicando métodos computacionais e estatísticos para treinar os modelos. Por fim, ressalta-se a criação dos multiagentes treinados para deliberar sobre proposições parlamentares, que utilizam métricas de desempenho, validação de modelos, previsões e classificações.

4.1.2 Natureza

Quanto à Natureza, as pesquisas podem ser classificadas em dois tipos, segundo Fonseca (2002):

- Pesquisa Básica: Objetiva gerar conhecimentos novos, úteis para o avanço da Ciência, sem aplicação prática prevista. Envolve verdades e interesses universais, e
- Pesquisa Aplicada: Objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.

No que se refere a este trabalho, foi realizada uma Pesquisa Aplicada, uma vez que foi desenvolvido um sistema multiagentes que visa simular as votações das proposições e obter resultados próximos da realidade, baseados em votações anteriores com temas semelhantes.

4.1.3 Objetivos

Para Gil (2007), com base nos objetivos, é possível classificar as pesquisas em três grupos:

- Pesquisa Exploratória: Este tipo de pesquisa tem como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a construir

hipóteses. A grande maioria dessas pesquisas envolve: (a) levantamento bibliográfico; (b) entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; e (c) análise de exemplos que estimulem a compreensão (GIL, 2007);

- Pesquisa Descritiva: A pesquisa descritiva exige do investigador uma série de informações sobre o que deseja pesquisar. Esse tipo de estudo pretende descrever os fatos e fenômenos de determinada realidade (TRIVIñOS, 1987), e

- Pesquisa Explicativa: Este tipo de pesquisa preocupa-se em identificar os fatores que determinam ou que contribuem para a ocorrência dos fenômenos Gil (2007). Ou seja, este tipo de pesquisa explica o porquê das coisas através dos resultados oferecidos. Segundo Gil (2007), uma pesquisa explicativa pode ser a continuação de outra descritiva, posto que a identificação de fatores que determinam um fenômeno exige que este esteja suficientemente descrito e detalhado.

  Pesquisas desse tipo podem ser classificadas como experimentais e ex-postfacto (GIL, 2007).

Nesse trabalho, foi conduzida uma pesquisa predominantemente Explicativa, pois buscou-se não apenas descrever os dados parlamentares, mas também entender e modelar as relações causais que podem influenciar as decisões dos agentes. Há um viés complementar, que pode ser visto como de cunho exploratório.

4.1.4 Procedimentos

De acordo com Fonseca (2002), a pesquisa possibilita uma aproximação e um entendimento da realidade a investigar, como um processo permanentemente inacabado. Processa-se através de aproximações sucessivas da realidade, fornecendo subsídios para uma intervenção no real.

Para se desenvolver uma pesquisa, é indispensável selecionar o método de pesquisa a utilizar. De acordo com as características da pesquisa, poderão ser escolhidas diferentes modalidades de pesquisa, sendo possível aliar o qualitativo ao quantitativo.

Nesse trabalho, foi empregada a pesquisa bibliográfica, uma vez que foi realizado um levantamento de artigos e livros com temas de interesse para o tema proposto. Dada a evolução da pesquisa, também é possível perceber a utilização da pesquisa-ação, uma vez que a coleta de dados, implementação do sistema e análise de resultados implica na participação do pesquisador.

4.2 Método Investigativo

Inicialmente, foi realizada uma pesquisa bibliográfica. Segundo Fonseca (2002), a pesquisa bibliográfica é feita a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de web sites. Qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto.

A pesquisa dos artigos foi realizada em algumas bases de dados, conforme listado no Quadro 3, utilizando-se de *strings* de busca para obter os resultados mais relevantes para o tema desse trabalho.

Dada a grande quantidade de resultados, optou-se por selecionar os artigos com o maior número de citações em artigos relacionados, considerando que artigos altamente citados geralmente indicam que a pesquisa é de alta qualidade e que seu conteúdo é relevante para a área de estudo.

Quadro 3 – *Strings* de busca utilizadas

String	**Base de dados**	**Resultados**
"Multiagent systems"	CAPES	10.290
"Multi-agent Systems AND Data Analytics"	CAPES	328
"Deep learning" AND "Multiagents systems"	CAPES	1.648
"Machine learning" AND "Multiagents systems"	CAPES	240
"Análise de sentimentos com redes neurais"	Google Acadêmico	21.400
"Neural Information Processing Systems"	Google Acadêmico	4.200.000
"Multiagents for Intelligent Decision"	Google Acadêmico	20.300
"Como são criadas as leis"	Google Acadêmico	318.000

Fonte: Próprio Autor

4.2.1 Critérios de Seleção

A seleção do material bibliográfico ocorreu com base na análise dos resumos e do capítulo de introdução dos artigos e da identificação das palavras-chave, onde buscou-se termos relacionados ao tema proposto. Dos critérios, cabe mencionar:

- Aborda o desenvolvimento de sistemas multiagentes;
- Utilização de redes neurais para o aprendizado dos sistemas multiagentes;

- Utilização de redes neurais para análise de sentimentos;
- Criação de sistemas multiagentes com JADE, e
- Processo legislativo na Câmara dos Deputados.

  Como resultados, têm-se o levantamento dos conceitos que embasam essa pesquisa, acordados no Capítulo 2 - Referencial Teórico, e as tecnologias, acordadas no Capítulo 3 - Referencial Tecnológico.

4.2.2 Principais Trabalhos Selecionados

Os trabalhos selecionados para este estudo abrangem diferentes abordagens e metodologias aplicadas à análise e processamento de dados por meio de sistemas multiagentes e redes neurais. Dos principais trabalhos selecionados para esse estudo, cabe mencionar:

- ***The Neural Network Zoo***: É apresentada uma visão geral das arquiteturas de redes neurais. Algumas dessas arquiteturas foram criadas nos últimos anos, enquanto outras têm origem há muitas décadas. Além de fornecer uma ferramenta prática para comparar modelos de *deep learning*, o Neural Network Zoo também revela uma taxonomia das arquiteturas de redes, sua cronologia e traça as linhagens e inspirações desses sistemas de processamento de informação neural (The Asimov Institute, Acesso em: 10 de abr. 2024);
- ***The Design and Implement of E-government information system based on Knowledge Management***: Diante dos problemas existentes no sistema de informação do governo eletrônico atual, este artigo propõe um sistema de governo eletrônico baseado na gestão do conhecimento. Foi apresentada a proposta de design geral da construção da plataforma. As características da gestão do conhecimento e a função de cada componente na estrutura foram analisadas. Um algoritmo de recomendação de informações foi projetado. A gestão, organização, mineração e recomendação personalizada de informações do governo eletrônico foram implementadas. Os resultados experimentais mostram que a plataforma possui boa escalabilidade e valor prático de aplicação (YUAN-YUAN; YONG-CHENG; HONG-MEI, 2011);
- ***Towards an Adaptive Multi-Agent System for Dynamic Big Data Analytics***: Uma tecnologia que se mostrou particularmente relevante para modelar, simular e resolver problemas em sistemas complexos são os Sistemas Multiagentes. Este artigo busca explorar e descrever como essa tecnologia pode ser aplicada ao *big data* por meio de um Sistema Multiagente Adaptativo, oferecendo capacidades analíticas dinâmicas. Essa pesquisa em andamento apresenta resultados promissores, mas

ainda precisa ser discutida e validada. Atualmente, está sendo aplicada no projeto *neOCampus*, o campus inteligente da Universidade de Toulouse III (BELGHACHE; GEORGÉ; GLEIZES, 2016);

- ***Multi-Agents Based Data Mining for Intelligent Decision Support Systems***: Neste artigo, é proposto uma técnica aprimorada de mineração de dados e sistemas multiagentes chamada DMMA, que utiliza uma abordagem de mineração em tempo real para analisar grandes volumes de dados em um ambiente distribuído. O estudo demonstrou que a velocidade de processamento foi aprimorada devido à abordagem de mineração multiagente, embora haja uma perda marginal de precisão. No entanto, essa diferença de precisão tende a diminuir com o tempo à medida que mais dados se tornam disponíveis (SHARMA; SHADABI, 2014), e

- ***Learning Long-Term Dependencies with Gradient Descent is Difficult***: As redes neurais recorrentes podem ser utilizadas para mapear sequências de entrada em sequências de saída, como em problemas de reconhecimento, produção ou previsão. No entanto, dificuldades práticas têm sido relatadas no treinamento dessas redes para tarefas em que as contingências temporais presentes nas sequências de entrada/saída abrangem longos intervalos. É demonstrado por que os algoritmos de aprendizado baseados em gradiente enfrentam desafios cada vez maiores à medida que a duração das dependências a serem capturadas aumenta. Esses resultados revelam um *trade-off* entre o aprendizado eficiente por descida do gradiente e a retenção de informações por longos períodos. Com base na compreensão desse problema, são consideradas alternativas ao método padrão de descida do gradiente. (BENGIO; SIMARD; FRASCONI, 1994);

  Como resultados, têm-se o levantamento dos conceitos que embasam essa pesquisa, acordados no Capítulo 2 - Referencial Teórico, e as tecnologias, acordadas no Capítulo 3 - Referencial Tecnológico.

4.3 Método de Desenvolvimento

Dada a necessidade de gerenciar e coordenar as atividades que serão desenvolvidas durante a criação desse trabalho, o autor optou por combinar duas metodologias ágeis: *Scrum* e *Kanban*.

Scrum é uma abordagem de gerenciamento de projetos que funciona de acordo com desenvolvimentos iterativos e incrementais. As etapas do modelo *Scrum* são, de acordo com Hossain, Babar e Paik (2009):

1. A criação do *backlog* do produto;

2. Planejamento das *sprints* e da criação do *backlog* das *sprints*;

3. Trabalho nas *sprints*, o software será desenvolvido como incrementos nas *sprints*, onde cada *sprints* dura duas semanas;

4. Reuniões de 15 minutos, que são realizadas todos os dias com três perguntas principais: o que foi feito no dia anterior, o que será feito hoje, e quais desafios imediatos devem ser enfrentados;

5. Teste de desempenho após cada *sprint*, e

6. Retrospectiva e planejamento das *sprints* futuras são realizados.

Concomitantemente, foi utilizado o *Kanban*, cujas principais vantagens são, segundo Ahmad, Markkula e Oivo (2013).:

1. Melhoria do tempo de entrega necessário para entregar o produto;

2. Melhoria da qualidade do produto;

3. Ênfase em coordenação e comunicação entre os membros da equipe, e

4. Obtenção de uma entrega consistente de produtos.

4.4 Método de Análise de Resultados

Como citado na seção Procedimentos, foi utilizado o método de pesquisa-ação. O objetivo é, segundo Engel (2000), unir a pesquisa à ação ou prática, isto é, desenvolver o conhecimento e a compreensão como parte da prática. Trata-se, portanto, de uma maneira de se fazer pesquisa em situações em que também se é uma pessoa da prática e se deseja melhorar a compreensão desta.

O método de pesquisa-ação é cíclico. As fases finais são usadas para aprimorar os resultados das fases anteriores. Nesse sentido, o método possui as seguintes fases, segundo Engel (2000):

- **Definição do problema**: Identifica-se todos os aspectos do problema com o qual se deseja trabalhar. A identificação do problema geralmente resulta de um período prévio de observação e reflexão, onde o pesquisador analisa o ambiente, define o foco da pesquisa e orienta as ações subsequentes;

- **Pesquisa preliminar**: Engloba os aspectos da revisão bibliográfica, afim de se o entender profundamente o problema que está sendo analisado;

- **Desenvolvimento de um plano de ação**: Realiza a elaboração de estratégias específicas para abordar o problema identificado e melhorar os processos ou resultados;
- **Coleta de dados para avaliação dos efeitos da implementação do plano**: defini-se sobre indicadores de avaliação e critérios de sucesso, seguidos pelas observações diretas para coletar dados quantitativos e qualitativos. Esses dados são analisados comparando-os com a linha de base para identificar padrões e tendências, e os resultados são documentados em um relatório detalhado, acompanhado de recomendações para ajustes no plano de ação conforme necessário, e
- **Comunicação dos resultados**: Divulga-se os principais achados e recomendações, além de uma análise detalhada dos dados coletados. Os resultados são discutidos e interpretados no contexto do problema original, destacando o impacto das ações implementadas.

4.4.1 Fluxo de Atividades

As diversas atividades que envolvem a elaboração desse trabalho estão divididas em duas etapas, sendo elas:

4.4.1.1 Primeira Etapa do TCC

A primeira etapa do Trabalho de Conclusão de Curso (TCC) visou-se estabelecer uma base sólida para o desenvolvimento do projeto. Esta etapa iniciou-se com a pesquisa bibliográfica, seguindo da definição do tema de pesquisa. Em seguida, foi desenvolvido o referencial teórico e tecnológico, onde se revisou a literatura existente e as tecnologias relacionadas ao tema escolhido, garantindo um embasamento sólido para a pesquisa. A metodologia também foi definida nessa fase, detalhando os métodos e técnicas utilizados para a coleta e análise dos dados, assegurando a validade e a confiabilidade dos resultados. Por fim, a proposta do TCC foi elaborada, apresentando os objetivos, justificativa e hipóteses do projeto que seria desenvolvido.

A Figura 3 evidencia as atividades que foram desenvolvidas na primeira etapa, cujo detalhamento é apresentado a seguir:

- **Definir Tema**: Essa atividade foi o ponto de partida do TCC, e teve como objetivo identificar áreas de interesse relevantes para o campo de estudo e viáveis para pesquisa. Durante essa etapa, discutiu-se sobre vários possíveis temas até se chegar ao tema definitivo da pesquisa;
- **Definir a Questão de Pesquisa**: Essa atividade abordou a questão de pesquisa que deve ser específica, clara e possível de ser respondida dentro do escopo do

trabalho. Ela orientou todo o processo de investigação, ajudando a manter o estudo direcionado e relevante;

- **Levantamento Bibliográfico**: Nesse subprocesso, foi realizado o levantamento bibliográfico, que envolveu a busca e a coleta de material acadêmico e científico relacionado ao tema e à questão de pesquisa;
- **Detalhar o Referencial Teórico**: Essa atividade permitiu identificar teorias, modelos e conceitos que sustentaram o estudo. Ela forneceu a base conceitual que guiou a interpretação dos dados e os resultados da pesquisa;
- **Detalhar o Referencial Tecnológico**: Essa atividade envolveu a descrição das ferramentas, técnicas e inovações tecnológicas abordadas ou utilizadas na pesquisa;
- **Detalhar a Metodologia de Pesquisa**: Essa atividade tratou da definição da metodologia de pesquisa, que foi elaborada para descrever os métodos e técnicas empregados ao longo do projeto. Isso incluiu métodos para pesquisa investigativa, método de desenvolvimento e método de análise de resultados;
- **Elaborar Proposta**: Essa atividade apresentou o plano detalhado do trabalho em andamento, como a definição do escopo, contexto e detalhes relacionados ao sistema proposto, e
- **Apresentar Primeira Etapa do TCC**: Essa atividade consistiu em apresentar todo o planejamento e os resultados obtidos da primeira etapa para a banca examinadora.

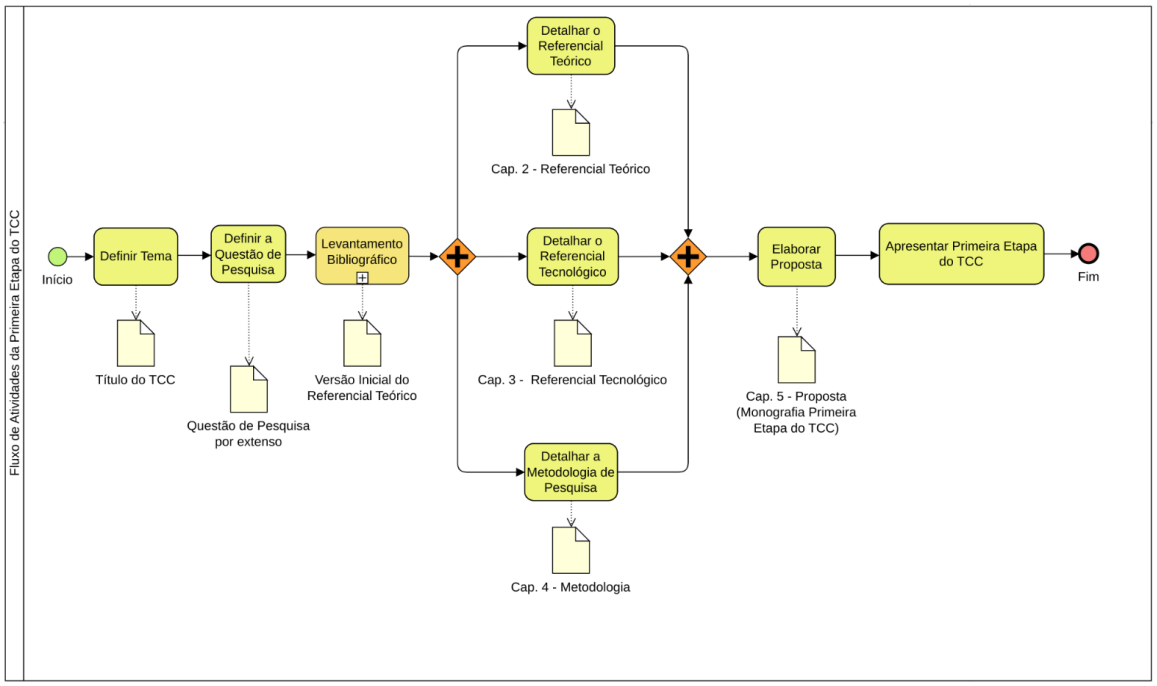
4.4.1.2 Segunda Etapa do TCC

Na segunda etapa, foram realizadas algumas correções e melhorias no trabalho, com base nas observações realizadas pela banca examinadora durante a apresentação da primeira etapa. Em seguida, foi iniciado o desenvolvimento do sistema proposto e a análise dos resultados com base nas métricas definidas.

Essas atividades estão evidenciadas na Figura 4, e são detalhadas a seguir:

- **Realizar Correções Sugeridas**: Nessa atividade, o *feedback* recebido da banca examinadora foi analisado e as correções sugeridas foram implementadas. Isso incluiu ajustes na estrutura do texto, correção de erros gramaticais, aprimoramento de argumentos e inclusão de informações adicionais necessárias;
- **Preparar Base de Registros para Treinamento**: Nessa atividade, as informações presentes na base de registros foram coletadas e organizadas de forma que somente os dados de interesse seguissem para o treinamento do modelo;

Figura 3 – Fluxo de atividades da primeira etapa do TCC



Fonte: Próprio Autor

- **Adicionar Eventuais Tópicos à Monografia**: Nessa atividade, todo o conteúdo da monografia foi revisado e foram identificados quaisquer lacunas ou tópicos adicionais que precisavam ser incluídos para garantir que o trabalho fosse completo e abrangente;

- **Criação do Projeto de Sistema Multiagentes**: Nesse subprocesso, foi desenvolvido o projeto de sistema multiagentes, incluindo a arquitetura do sistema, a definição dos agentes, suas funções e interações. Também foram realizados testes para garantir que o sistema multiagentes funcionasse corretamente e atendesse aos objetivos propostos;

- **Treinamento da Rede Neural**: Nesse subprocesso, foi realizada a configuração da arquitetura da rede LSTM, incluindo a alimentação dos dados de treinamento, ajustes dos pesos através de otimização, validação do desempenho e número de iterações até que o modelo atingisse um desempenho satisfatório;

- **Adição de Comportamento aos Agentes**: Nesse subprocesso, foram programadas as ações e reações dos agentes conforme regras definidas, implementando esses comportamentos no sistema e realizando testes para garantir que eles funcionassem conforme esperado;

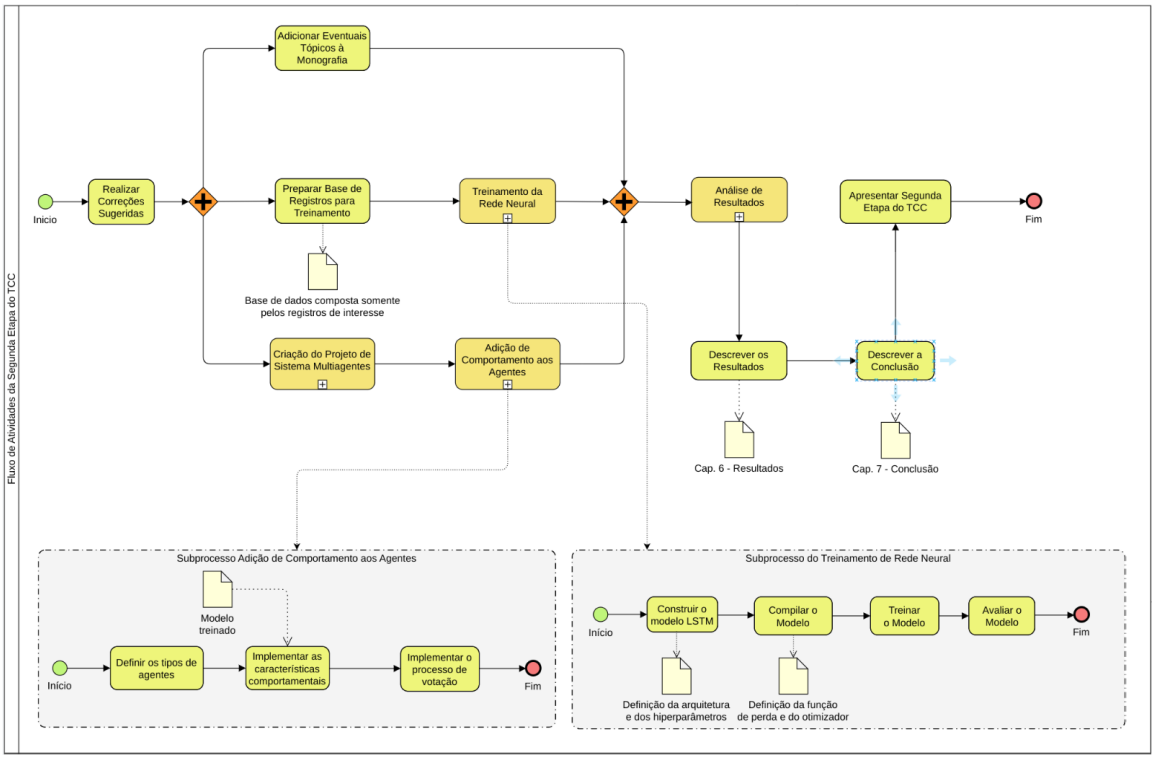
- **Análise de Resultados**: Subprocesso que envolveu a coleta e a análise dos dados gerados pelos experimentos ou simulações;

- **Descrever os Resultados**: Nessa atividade, foram documentadas detalhadamente as informações obtidas, apresentando os dados de maneira clara e interpretando os resultados;

- **Descrever a Conclusão**: Essa atividade envolveu sintetizar os principais resultados e contribuições do trabalho, destacando sua relevância, apontando limitações e sugerindo direções para pesquisas futuras, e

- **Apresentar Segunda Etapa do TCC**: Nessa atividade, em andamento, tem-se o preparo e a apresentação de todo o progresso do trabalho desde a primeira etapa para a banca examinadora.

Figura 4 – Fluxo de atividades da segunda etapa do TCC



Fonte: Próprio Autor

4.5 Cronograma das Atividades

Nesta seção, é apresentado o cronograma referente a todas as atividades realizadas ao longo do tempo estipulado para a conclusão da primeira etapa do TCC - Quadro 4, e para a segunda etapa - Quadro 5.

Ressalta-se que todas as atividades e/ou os subprocessos da primeira etapa foram realizados, incluindo a atividade Apresentar Primeira Etapa do TCC. Adicionalmente, as

atividades e/ou subprocessos da segunda etapa foram concluídas, com exceção da atividade de Apresentar Segunda Etapa do TCC, e os demais insumos encontram-se apresentados no Capítulo 5 - Sistema Multiagente de Simulação de Votação Parlamentar.

Quadro 4 – Cronograma das atividades relacionadas à primeira etapa do TCC

Atividade/Subprocesso	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**
Definir Tema	X				
Definir a Questão de Pesquisa	X	X			
Levantamento Bibliográfico		X	X		
Detalhar o Referencial Teórico		X	X		
Detalhar o Referencial Tecnológico		X	X		
Detalhar a Metodologia de Pesquisa			X	X	
Elaborar Proposta				X	
Apresentar Primeira Etapa do TCC					X

Fonte: Próprio Autor

Quadro 5 – Cronograma das atividades relacionadas à segunda etapa do TCC

Atividade/Subprocesso	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**
Realizar Correções Sugeridas	X						
Preparar Base de Registros para Treinamento	X						
Adicionar Eventuais Tópicos à Monografia		X					
Criação do Projeto de Sistema Multiagentes		X					
Treinamento da Rede Neural			X		X	X	
Adição de Comportamento aos Agentes			X	X			
Análise de Resultados					X	X	
Descrever os Resultados					X	X	
Descrever a Conclusão						X	X
Apresentar Segunda Etapa do TCC							X

Fonte: Próprio Autor

4.6 Considerações Finais do Capítulo

Esse capítulo tratou das questões metodológicas que nortearam essa pesquisa. Foram apresentados os aspectos que classificam essa pesquisa; a forma como ocorreu o levantamento bibliográfico, e um detalhamento sobre as *strings* de pesquisa. Também foram apresentadas as metodologias ágeis adotadas para o desenvolvimento desse projeto, tais como o *Scrum* e o *Kanban*. Ainda tratando da metodologia, foi apresentada a forma como análise dos resultados foi conduzida, a qual aplica o método de pesquisa-ação.

Finalmente, todas as atividades realizadas e planejadas para as duas fases do Trabalho de Conclusão de Curso foram descritas, incluindo seus cronogramas que detalham as atividades e o período de execução.

5 Sistema Multiagentes para Simulação de Votação Parlamentar

Este capítulo tem por finalidade apresentar a Contextualização, que aborda informações relacionadas ao problema em questão e ao desenvolvimento do sistema. Adicionalmente, na Base de Dados, há detalhes quanto à obtenção dos dados públicos provenientes da base de dados da Câmara dos Deputados, e ao tratamento destes dados para que fossem utilizados no treinamento das redes neurais. Em seguida, têm-se as Redes Neurais, que acordam sobre as redes reunais utilizadas; o processo de treinamento, e as métricas aferidas na avaliação dos resultados. Adiante, é apresentado o Desenvolvimento dos Agentes, que define quais os tipos de agentes utilizados, e as características de cada um. Por fim, são apresentadas as Considerações Finais do Autor, com um panorama geral do que foi discutido neste capítulo.

5.1 Contextualização

Observando a crescente quantidade de dados legislativos gerados diariamente, percebe-se que a análise e a deliberação sobre proposições parlamentares são tarefas complexas e desafiadoras. Tecnologias emergentes como sistemas multiagentes, aprendizado de máquina e aprendizado profundo oferecem a possibilidade de simular esses processos. Este trabalho apresenta um projeto de sistema multiagentes, com agentes treinados usando técnicas de aprendizado de máquina e aprendizado profundo para analisar proposições parlamentares e simular deliberações.

A adoção de tecnologias avançadas na análise legislativa proporcionam maior eficiência, precisão e transparência no processo de deliberação parlamentar. Sistemas multiagentes podem simular comportamentos de diferentes atores políticos, enquanto algoritmos de aprendizado de máquina identificam padrões e tendências nas proposições. Esta abordagem tem o potencial de apoiar legisladores e analistas, oferecendo percepções valiosas e previsões fundamentadas em dados.

Tendo em vista o exposto, o autor realizou uma análise sobre a base de dados citada e a aplicação de recursos informáticos, criando um sistema multiagente que apresenta uma visão baseada em padrões sobre o processo de votação, possibilitando a comparação dos resultados obtidos na simulação com os dados das votações reais.

No presente trabalho, considerou-se a utilização de Redes Neurais de Propagação Direta, tais como as Redes Convolucionais, *Autoencoders*, Redes de Atenção e Redes Con-

volucionais e de Redes Neurais Recorrentes, como a de Memória de Curto e Longo Prazo. Entretanto focou-se no uso das Redes Neurais Recorrentes, em específico, na Memória de Curto e Longo Prazo, cujas implementações são providas pelo *Deeplearning4j*. Isso ocorreu, uma vez que essas implementações são particularmente eficazes no processamento e na classificação de dados em formato de texto, conforme observado para o caso das proposições e dos discursos dos deputados. Além disso, considerou-se análises de séries temporais, tais como o histórico de votos ao longo do tempo.

O sistema desenvolvido, o qual foi hospedado no *Github*, encontra-se disponível para acesso via o link: <https://github.com/DouglasMonteles/politics-agents>.

5.2 Base de Dados

A base de dados utilizada é a de dados públicos sobre a Câmara dos Deputados, portal que foi lançado oficialmente em 2017, e que surgiu devido à Lei de Acesso à Informação, promulgada em 2011 (Câmara dos Deputados, 2024).

Esse portal fornece dados dos mais diversos tipos. Sendo assim foi necessário selecionar os tipos de dados mais interessante para o escopo desse projeto. Nesse sentido, os tipos de dados selecionados foram:

- **Dados dos Deputados**: Lista de deputados que estiveram em exercício parlamentar em algum intervalo de tempo, onde esses dados são utilizados para compor as características de cada agente deputado, tais como o seu nome e o partido ao qual pertence. A estrutura dos Dados dos Deputados encontra-se apresentada no Apêndice A;
- **Dados dos Partidos**: Partidos políticos que têm ou já tiveram deputados na Câmara. A estrutura dos Dados dos Partidos encontra-se apresentada no Apêndice B;
- **Votos dos Deputados**: Voto ou posicionamento de um deputado, que foi registrado em uma votação nominal que não tenha sido secreta. Com isso, pode-se identificar o voto individual de cada deputado e realizar um comparativo com o resultado obtido pelo agente de software. A estrutura dos Dados dos Votos de uma votação encontra-se apresentada no Apêndice C;
- **Votações**: Votações das comissões e do Plenário da Câmara, por ano em que tenham ocorrido. Para cada votação, há data informada; data e hora de registro nos sistemas da Câmara; descrição do resultado da votação; identificadores do Órgão e do Evento; indicador que diz se a votação resultou em aprovação ou não da proposição votada; placar resumido e, se disponíveis, dados sobre a abertura de votação e a apresentação

de proposições que tenham ocorrido logo antes da votação. A estrutura dos Dados das Votações encontra-se apresentada no Apêndice D;

- **Dados das Proposições**: Lista de informações básicas sobre projetos de lei, resoluções, medidas provisórias, emendas, pareceres e todos os outros tipos de proposições na Câmara. A estrutura dos Dados das Proposições encontra-se apresentada no Apêndice E, e
- **Orientações de Voto**: Em muitas votações, os líderes de partidos e blocos - as bancadas - fazem recomendações de voto para seus parlamentares. Essas orientações de uma votação também são feitas pelas lideranças de Governo, Minoria e as mais recentes Maioria e Oposição. A estrutura dos Dados das Orientações encontra-se apresentada no Apêndice F;

Com esses tipos de dados, foi possível determinar o voto de um deputado(a) com base na orientação do seu partido, sendo a posição do partido – a favor ou contra – o fator decisivo para a definição do voto durante a votação.

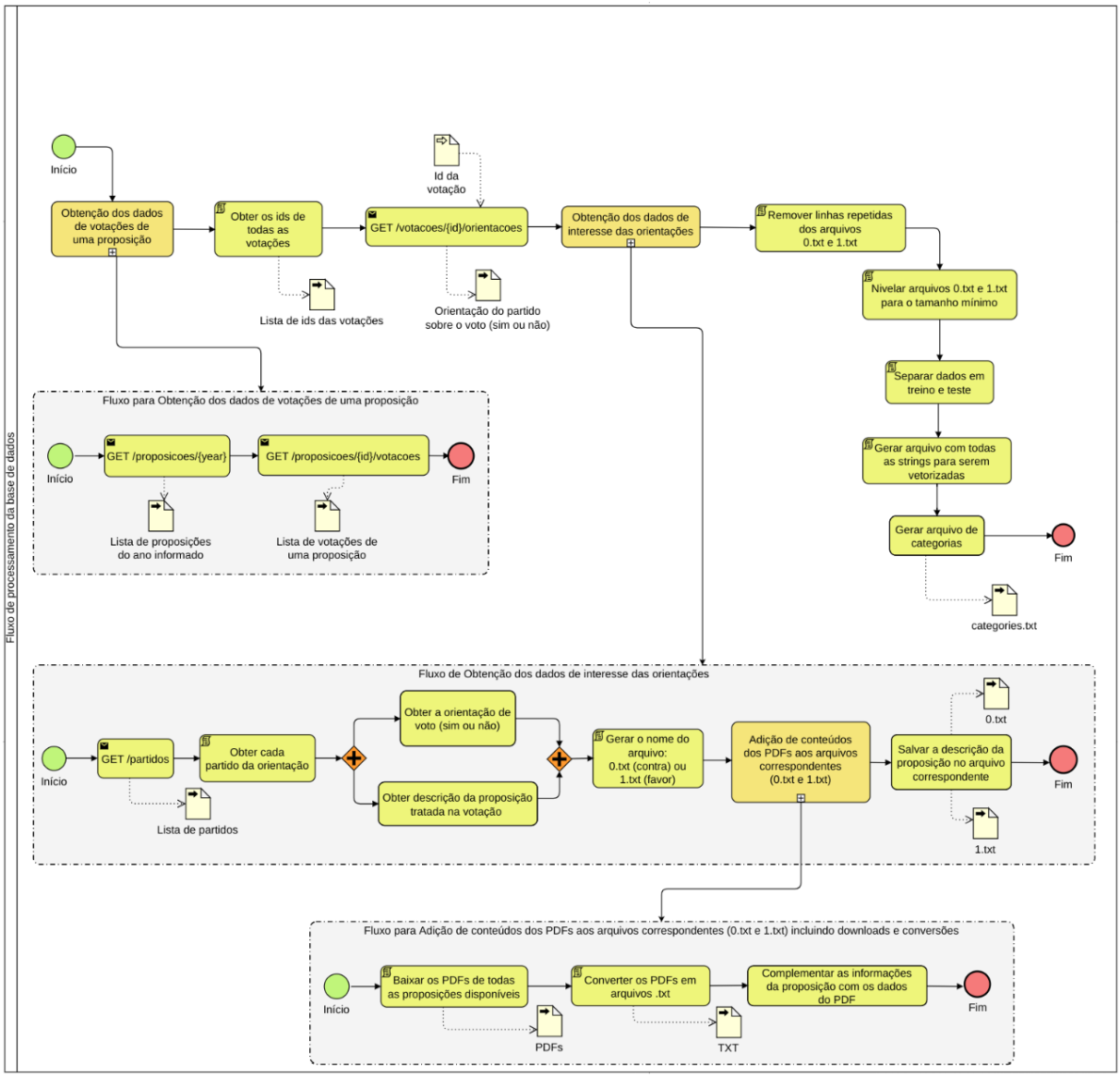
Para organizar o processo de *download* e preparação dos dados, foram criados dois módulos específicos no projeto com a lógica necessária para isso. O primeiro deles é o *external-data-processor*, que realiza os seguintes passos para a obtenção dos dados e preparação dos mesmos para o treinamento:

1. Realiza o *download* das proposições no intervalo dos anos de 2008 a 2023 e, para cada proposição, é realizado o *download* das informações de votação, quando existirem;
2. Realiza o *download* das orientações de um partido quanto ao voto de um deputado(a) em uma votação de uma determinada proposição, quando existir;
3. Realiza o *download* dos arquivos em PDF contendo mais informações sobre as proposições baixadas anteriormente. Depois, os converte em arquivos de texto puro;
4. Realiza o processo de gerar dois arquivos em texto puro, sendo eles *1.txt* (Votos a favor) e *0.txt* (Votos contra). Cada um desses arquivos possui um resumo da proposição votada;
5. Realiza o processamento desses arquivos para deixá-los prontos para o treinamento. Para isso, os arquivos *1.txt* e *0.txt* tem a quantidade de linhas niveladas para o menor número dentro os dois arquivos. Depois, os dados são separados em treino e teste, seguindo a proporção 80% para treino e 20% para testes;
6. Cria o arquivo *ProposalWordVector.txt*, e o preenche com o texto de todas as proposições, e

7. Cria o arquivo *categories.txt*, que lista as categorias dos arquivos. Para esse projeto, as categorias são *0* para votos contra, e *1* para votos a favor.

Esse processo está representado visualmente no diagrama BPMN da Figura 5.

Figura 5 – Fluxo de processamento dos dados para treinamento



Fonte: Próprio Autor.

O software desenvolvido possui dois módulos que tratam do processo de *download* os dados e processamentos das informações, para deixá-los prontos para serem utilizados no treinamento dos modelos. São eles:

- *external-data-processor*: Lida com todo o processo de acesso à base de dados da Câmara dos Deputados; *download* dos dados de interesse, e conversão dos mesmos para objetos Java. Depois, permite processar propriedades específicas, tais como a

descrição das proposições, e salvá-las em um arquivo de texto. A Figura 6 apresenta as principais classes desse módulo, e

- *conversor-pdf-to-plain-text*: Esse é um módulo auxiliar, escrito diretamente em Python, que permite converter os PDFs que contém informações detalhadas de uma proposição em texto puro. Essa estratégia permite um aumento na quantidade de dados disponíveis para treinamento do modelo de cada partido. A Figura 7 apresenta a estrutura do *script* desenvolvido.

Ressalta-se que existem interfaces padronizadas para acesso aos dados dos serviços dedicados, conforme consta em *DeputyService*, *ProposalService* e outras interfaces. Além disso, consta uma fachada, *PrepareDatabaseFacade*, a qual recebe requisições, e as encaminha para tratamento de cada parte do processo, como por exemplo, para: *downloadVotingProposals()* e *prepareDatabaseForTraining()*. Respectivamente, esses métodos cuidam do download das proposições votadas, bem como da preparação da base de dados para treino.

5.3 Redes Neurais

Dentre os diversos tipos de redes neurais analisados na seção 2.3 do Capítulo 2, utilizou-se uma Rede Neural Recorrente (RNR), pois lidam de forma eficaz com dados sequenciais e dependências de longo prazo, essenciais para analisar e interpretar discursos e votações parlamentares que possuem uma ordem temporal e lógica.

Com isso, criou-se um sistema que armazena e atualiza informações de contexto; ou seja, informações computadas a partir de entradas passadas e úteis para produzir as saídas desejadas (BENGIO; SIMARD; FRASCONI, 1994).

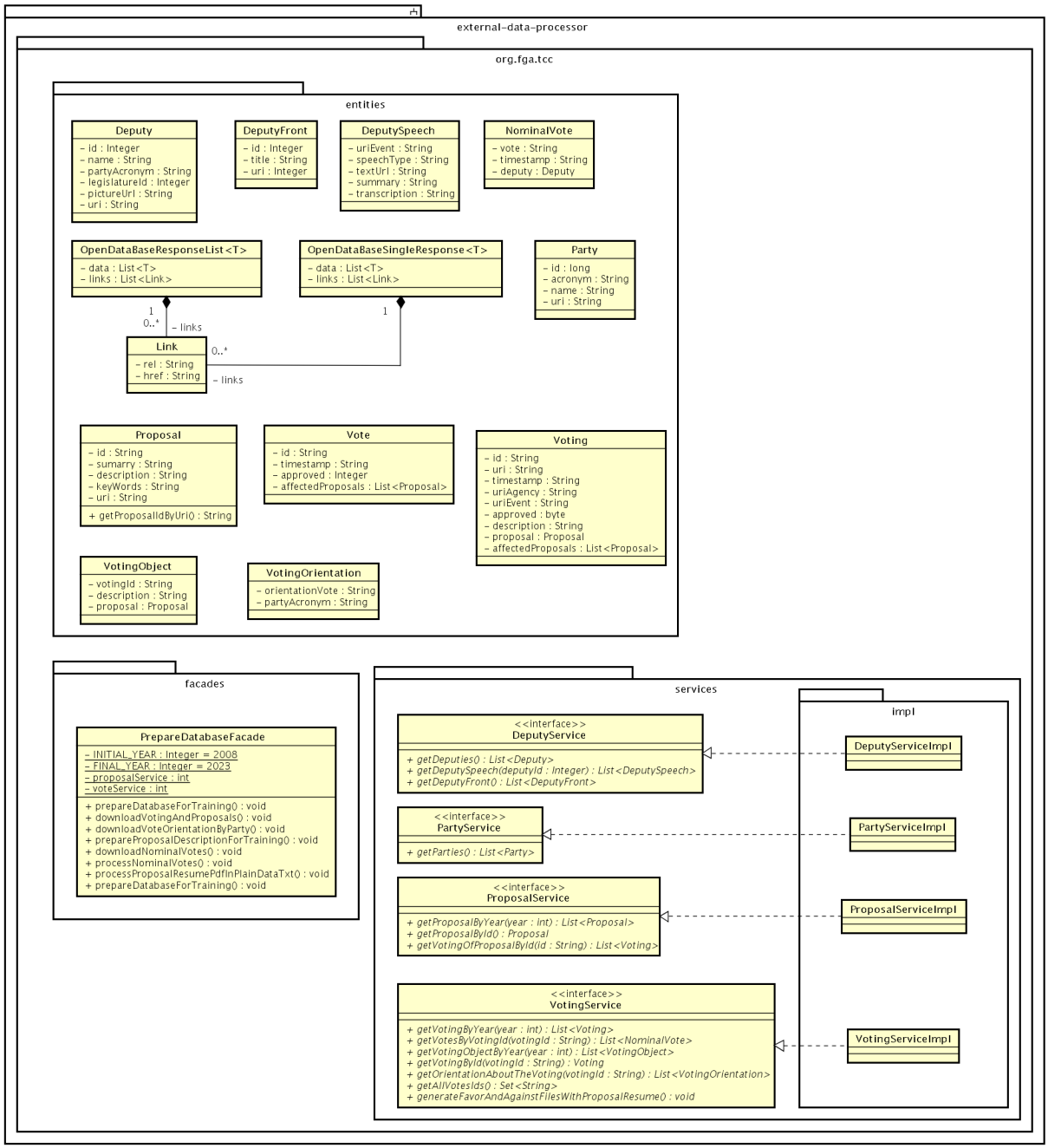
5.3.1 *Memória de Curto e Longo Prazo*

Para esse projeto, as LSTMs são utilizadas para analisar os discursos dos deputados e o histórico de votos, a fim de se identificar categorias, tópicos e posições políticas. Também é utilizada para prever como os deputados votarão em proposições com base nas orientações fornecidas pelos seus respectivos partidos em votações anteriores.

No contexto desse trabalho, o uso dessa rede neural é util para:

- Problema: Dado um conjunto de agentes, faz-se necessário treiná-los para que possam deliberar sobre proposições apresentadas à Câmara dos Deputados. Eles devem basear seu julgamento em casos semelhantes, que já foram discutidos e votados.

Figura 6 – Diagrama de classes do módulo de processamento da base de dados

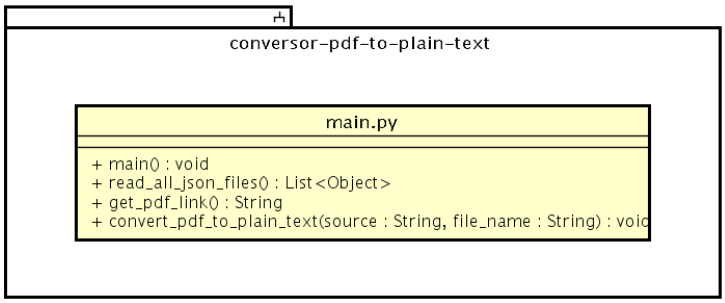


Fonte: Próprio Autor.

- Casos Recuperados: Os Dados Abertos da Câmara dos Deputados fornecem todas as proposições já votadas entre os anos de 2008 até 2023 (intervalo escolhido para trabalhar com dados). Adicionalmente, as informações não se limitam apenas proposições. Há ainda o fornecimento de informações públicas sobre as votações, os votos nominais, as orientações de votação de cada partido, entre outros, e

- Adaptação: Aplicação de técnicas de Processamento de Linguagem natural e Aprendizado profundo para conseguir definir o comportamento de cada agente com base na opinião ou no julgamento realizada(o) no passado e que está relacionada(o) à

Figura 7 – Diagrama de classes do módulo auxiliar de conversão de PDFs em texto



Fonte: Próprio Autor.

uma determinada proposição.

5.3.2 Treinamento

Para o treinamento, utilizou-se das descrições das proposições que foram obtidas de todas as votações que possuíam as orientações dos partidos sobre o voto a ser dado pelos seus deputados. A Figura 5 ilustra as inerentes atividades envolvidas para obtenção da descrição das proposições, bem como para armazenamento dessas informações em arquivos correspondentes ao voto que foi dado como orientação pelos partidos. Ainda nesse contexto, os arquivos são 0.txt, no qual cada linha possui a descrição das proposições cuja orientação do partido foi votar contra, e 1.txt, no qual cada linha possui a descrição das proposições cuja orientação do partido foi votar a favor.

Todo o processo de treinamento foi isolado no módulo *trained-lstm*, o qual possui todas as dependências do *Deeplearning4j* necessárias para geração, treinamento e teste do modelo. Os dados que foram utilizados no treinamento estão disponíveis no diretório *trained-data*. Nele, há diretórios para cada partido contendo os arquivos separados em diretórios de teste e treino. O treinamento engloba os seguintes passos:

1. Definição do caminho para o diretório do partido, onde cada partido tem um modelo treinado;

2. Geração do arquivo *WordVector.txt*, que armazena as palavras presentes no arquivo *ProposalWordVector.txt* convertidas em vetores;

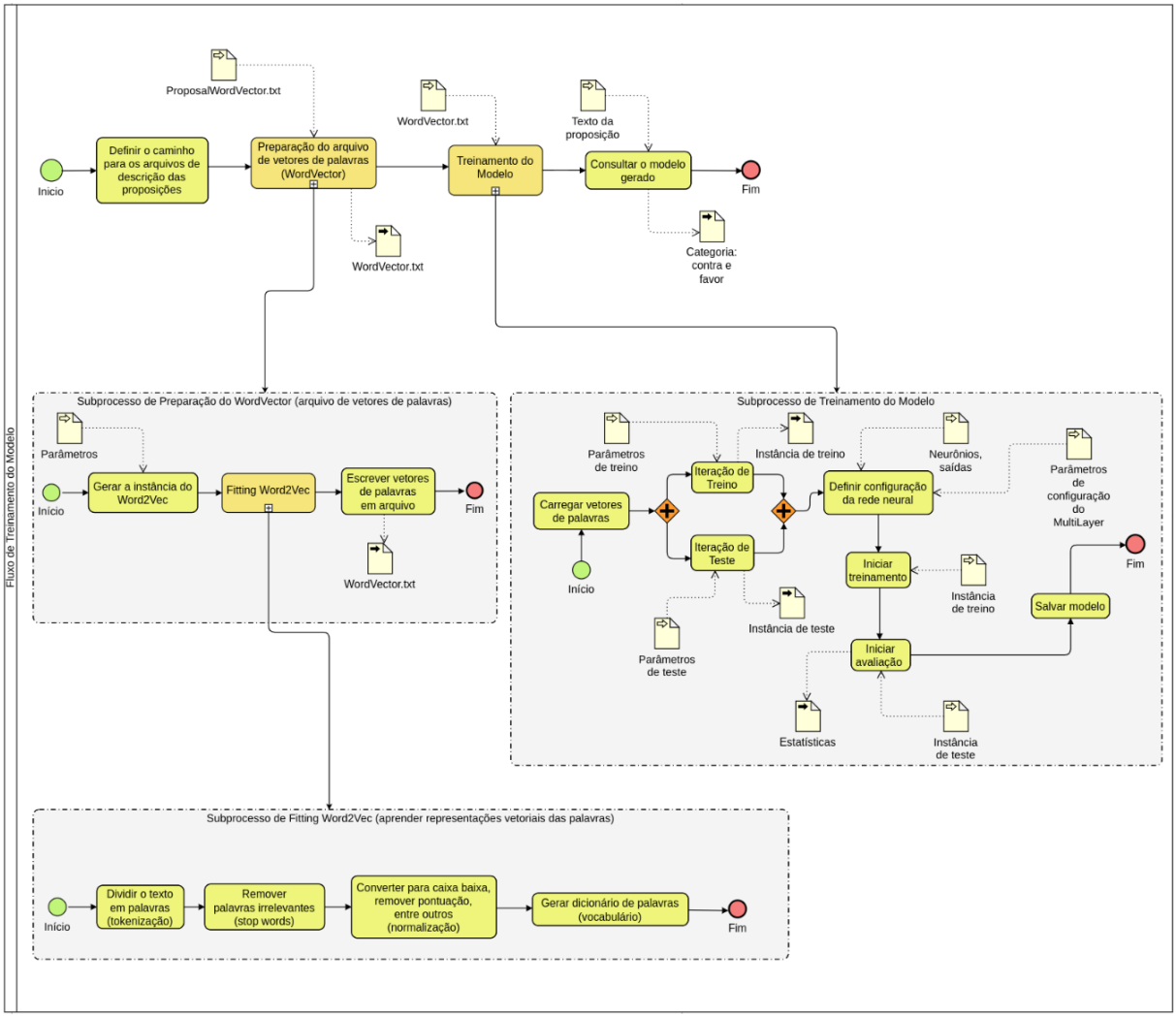
3. Treinamento do modelo, utilizando os dados de treino e definindo a configuração da rede neural;

4. Avaliação do modelo gerado utilizando os dados de teste, e

5. Registro do modelo gerado no diretório específico do partido.

Na Figura 8, tem-se a representação de todo o fluxo de definição, treinamento, avaliação e registro do modelo no diretório do partido.

Figura 8 – Fluxo de treinamento dos modelos de cada partido



Fonte: Próprio Autor.

5.3.3 Métricas

A fim de avaliar o resultado do treinamento dos modelos, é importante a definição de métricas que evidenciem como está o desempenho do modelo, garantindo que ele esteje atingindo os objetivos desejados.

Para esse trabalho, foram utilizadas as seguintes métricas:

- **Acurácia**: A acurácia, também chamada de precisão geral do modelo, é uma medida que corresponde à uma porcentagem de todas as predições realizadas corretamente por todos os valores obtidos. A acurácia pode ser calculada de acordo com o exposto na Equação (5.1) (JURAFSKY; MARTIN, 2019):

$$\text{acurácia} = \frac{VP + VN}{VP + FP + VN + FN} \tag{5.1}$$

- **Precisão**: A precisão é uma métrica que calcula a efetividade de uma predição positiva correta. A precisão pode ser obtida dividindo o total de verdadeiros positivos pela soma das predições verdadeiros positivos e falsos positivos, regida de acordo com o exposto na Equação (5.2) (JURAFSKY; MARTIN, 2019):

$$\text{precisão} = \frac{VP}{VP + FP} \tag{5.2}$$

- **Revocação**: Utiliza-se a revocação para entender a frequência com que o modelo consegue predizer os dados de uma classe corretamente. O cálculo da revocação é obtido dividindo os verdadeiros positivos pela soma de verdadeiros positivos com falsos negativos (JURAFSKY; MARTIN, 2019), de acordo com o exposto na Equação (5.3):

$$\text{revocação} = \frac{VP}{VP + FN} \tag{5.3}$$

- **Medida F1**: A Medida F1 é uma média ponderada harmônica que combina precisão e revocação. A medida consegue lidar com classes desbalanceadas atribuindo maior peso ao menor valor. O resultado é utilizado para avaliar a qualidade do modelo, onde o *score* máximo será uma pontuação igual (JURAFSKY; MARTIN, 2019), de acordo com o exposto na Equação (5.4):

$$MedidaF1 = 2 \cdot \frac{precisão \cdot revocação}{precisão + revocação} \tag{5.4}$$

- **Matriz de Confusão**: A matriz de confusão é uma tabela que descreve o desempenho de um modelo de classificação. Ela compara as classes reais (verdadeiras) com as classes previstas pelo modelo, mostrando o número de verdadeiros positivos (VP); falsos positivos (FP); verdadeiros negativos (VN), e falsos negativos (FN). Um exemplo de matriz de confusão é mostrado no Quadro 6.

Quadro 6 – Matriz de confusão

		Classe Real	
		Positivo	Negativo
Predição do Modelo	Positivo	VP	FP
	Negativo	FN	VN

Fonte: Próprio Autor

No contexto desse trabalho, a matriz de confusão representa os resultados do modelo de classificação que prevê votos parlamentares (contra = 0 ou a favor = 1) em relação ao conjunto de dados, e sua representação é apresentada no Quadro 7.

Quadro 7 – Matriz de confusão do modelo proposto

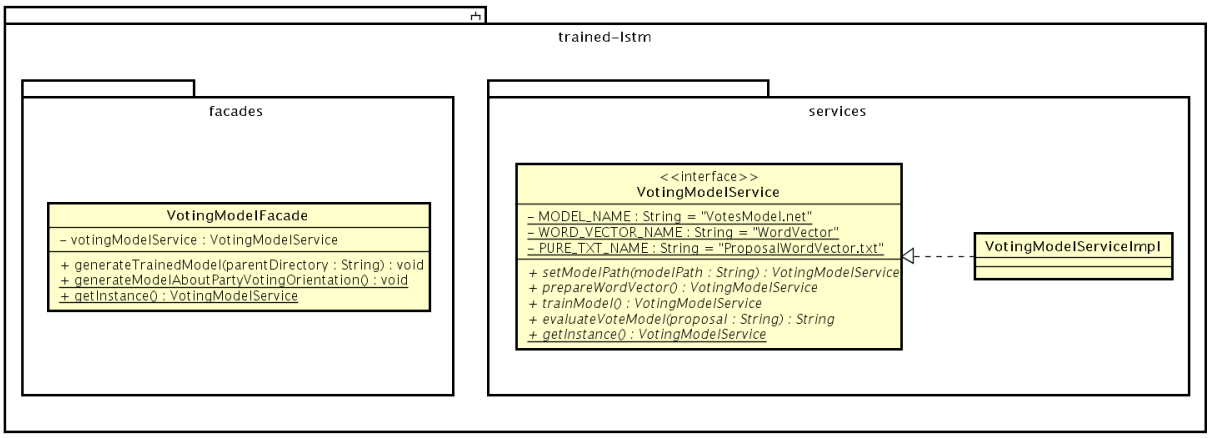
		Classe Real	
		A favor	Contra
Predição do Modelo	A favor	VP	FP
	Contra	FN	VN

Fonte: Próprio Autor

No software desenvolvido, todo o processo de treinamento também foi isolado em um módulo específico, sendo ele o *trained-lstm*, o qual possui acesso a todos os dados de treinamento e à lista de partidos. A Figura 9 apresenta as principais classes de módulo.

Novamente, o módulo está estruturado usando o padrão de projeto Fachada, com *VotingModelFacade*, para tratamento de requisições que demandam, por exemplo, gerar modelo treinado via o método *generateTrainedModel(...)*. Além disso, há uso da abstração da interface *VotingModelService*, padronizada, que auxilia na implementação concreta de VotingModelServiceImpl.

Figura 9 – Diagrama de classes do módulo de treinamento dos modelos dos partidos



Fonte: Próprio Autor.

5.4 Desenvolvimento dos Agentes

Um agente de software é conceituado como uma entidade que funciona de forma contínua e autônoma em um ambiente em particular, geralmente habitado por outros agentes, e que seja capaz de intervir no seu ambiente, de forma flexível e inteligente, sem requerer intervenção ou orientação humana constante (SILVEIRA, 2006).

Para fins deste projeto, os agentes planejados, e que foram desenvolvidos, estão separados por tipos, onde cada tipo atua em uma parte do processo de deliberação de proposições:

- **Agentes Participativos**: Esses agentes desempenham os papéis dos deputados,

herdando suas características, opiniões e ideologias. Esses agentes utilizam redes neurais LSTMs para prever votos com base nas orientações do partido sobre votações anteriores;

- **Agentes de Gerenciamento**: Esses agentes são responsáveis por iniciar os Agentes Participativos. Todo o processo de votação é gerenciado por eles, iniciando pela apresentação do texto da proposição; pela criação dos agentes que simulam os deputados(as), e pelo recebimento dos votos e propagação do resultado, e
- **Agentes de Interface**: Esses agentes interagem com o usuário final, fornecendo visualizações e relatórios sobre as análises e simulações realizadas pelos agentes.

Com acesso ao modelo treinado, os agentes podem desempenhar cada um destes papéis e interagir entre si para votarem em uma determinada proposição, considerando as especificidades de cada um.

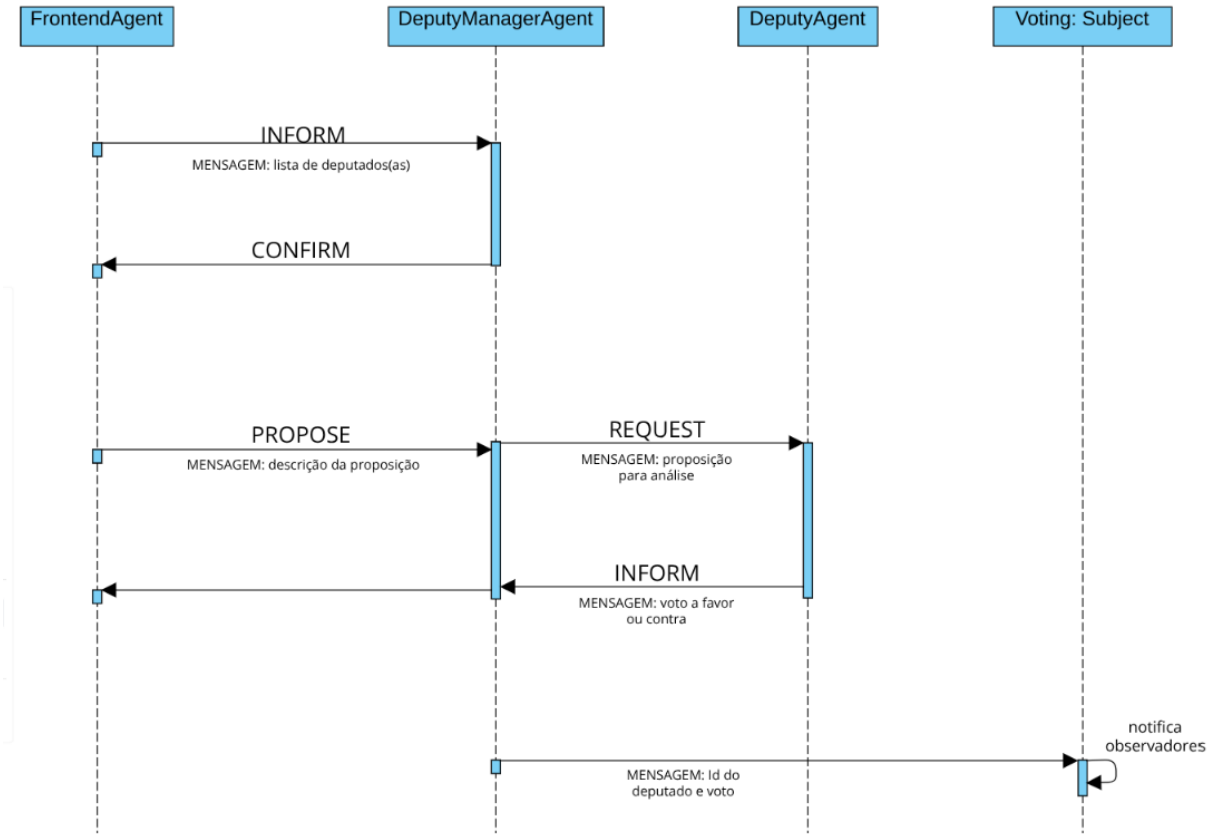
Segue, na Figura 10, o diagrama de sequência, evidenciando a comunicação entre os diferentes tipos de agentes citados. Percebe-se uma comunicação ponto a ponto entre o Agente de Interface e o Agente de Gerenciamento. O mesmo pode ser observado entre este e o Agente Participativo. Na Figura 11, visualiza-se a mesma interação entre os agentes, utilizando o *sniffer* do JADE. Trata-se de uma ferramenta gráfica, já oferecida pelo *framework* JADE, que permite visualizar as trocas de mensagens entre os agentes, compreendendo que essas trocas orientam-se pela performativas de mensagens da FIPA (ex. INFORM, CONFIRM, PROPOSE, dentre outras).

Por fim, o resultado da comunicação entre os agentes é o voto e a identificação do deputado(a), que por sua vez, é propagado a todos os observadores. No contexto desse projeto, os observadores são as interfaces criadas com *Java Swing*, que apresentam o resultado para o usuário, e precisam atualizar as informações constantemente.

Todos os agentes foram separados em um módulo específico, sendo ele o *agents*, cuja estrutura de classes está evidenciada na Figura 12. Uma parte interessante desse nível de modelagem é observar, no pacote *jade.core*, a entidade que representa o agente, ou seja, *Agent*, bem como o principal comportamento utilizado na programação dos agentes comportamentais em Jade, no caso, o *CyclicBehaviour*.

Consta ainda o uso de ontologias, conforme apresentado no pacote *ontologies*. Ontologias ajudam na semântica do sistema implementado, tendo como base o uso de termos que são mais compreensíveis para o domínio em estudo. Por exemplo, termos como: *proposal* e *deputy*. Além disso, mantêm-se as relações semânticas entre esses termos via um modelo semântico, no caso, evidenciado nas associações entre os pacotes *predicate* e *concept*. Esse cuidado é muito valorizado em SMAs, uma vez que esse paradigma enfatiza a necessidade de se representar o mais próximo possível as particularidades de um domínio.

Figura 10 – Fluxo de comunicação entre os agentes



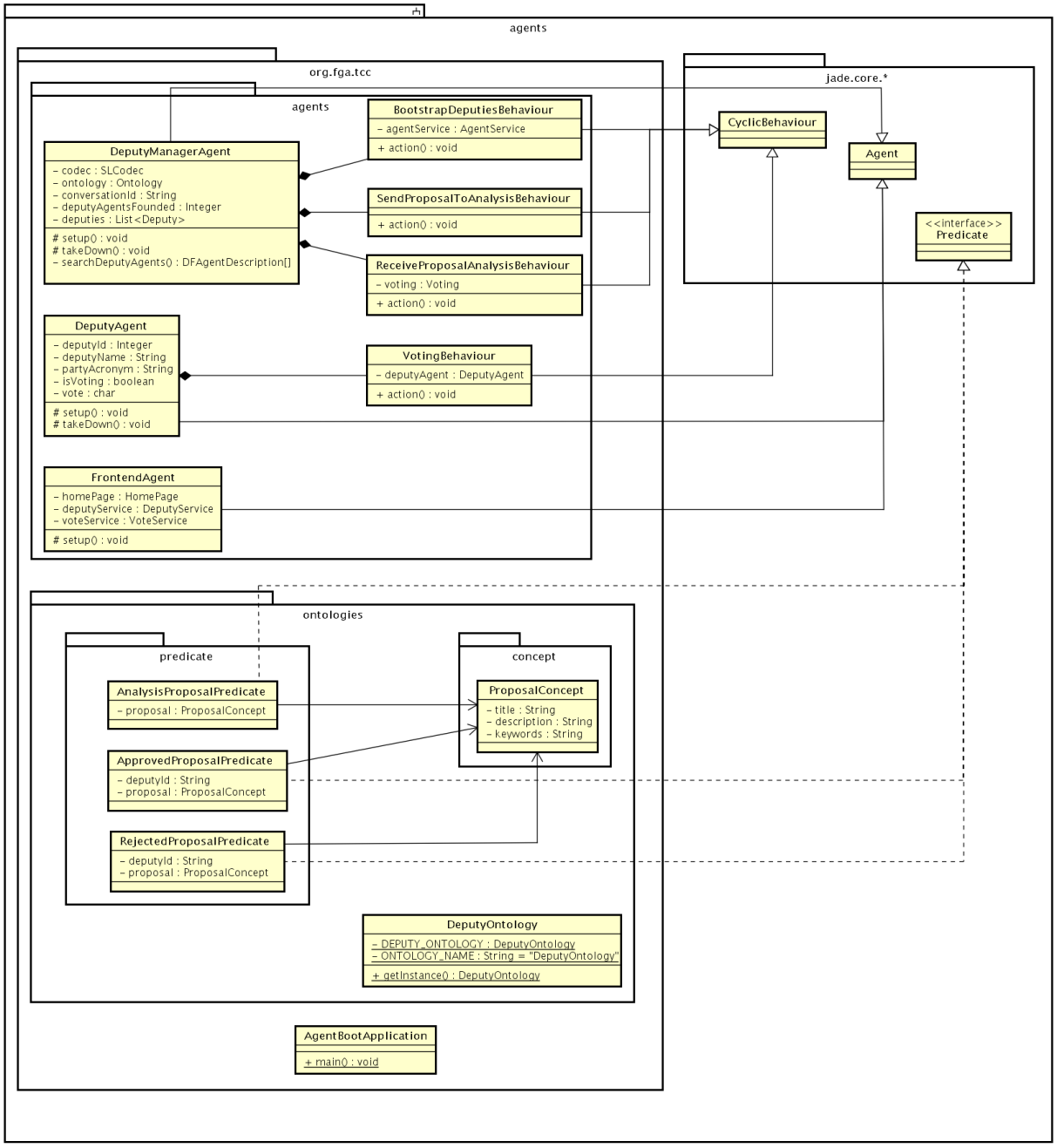
Fonte: Próprio Autor.

Figura 11 – Troca de mensagens entre os agentes

Fonte: Próprio Autor.

O projeto foi criado utilizando o *maven*, Figura 13, que facilita o processo de gerenciamento de dependências. Para a execução, é necessário ter a versão 21 do Java instalad.a Adicionalmente, foi criado, na raiz o projeto, um arquivo *Makefile*, para facilitar o processo de compilação e execução. Para executá-lo, basta abrir um terminal na raiz do projeto e executar o comando *make*. Ao final, a janela de gerenciamento de agentes do JADE deve ser exibida, como acontece na Figura 14.

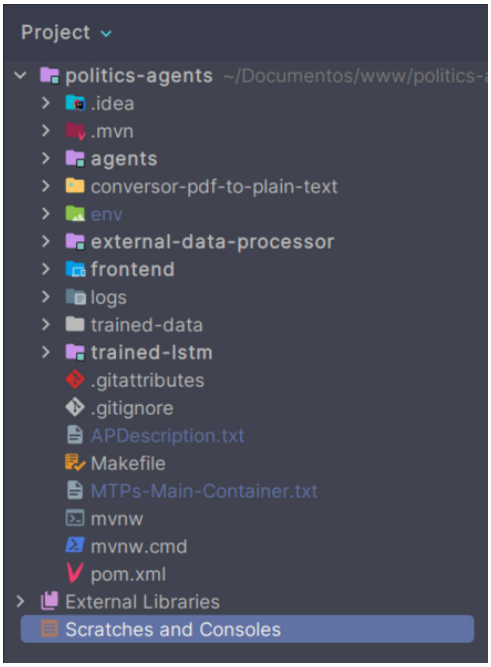
Figura 12 – Diagrama de classes do módulo de agentes



Fonte: Próprio Autor.

Para o gerenciamento dos agentes, cabe o uso de novos recursos gráficos já oferecidos pela plataforma JADE. Nesse caso, trata-se da interface gráfica que evidencia - via uma estrutura hierárquica, a plataforma, o *Main-Container*, e todos os agentes em execução, em um momento específico do sistema rodando. Importante ressaltar que há a presença dos principais agentes da plataforma JADE, que fazem parte da arquitetura do *framework*, no caso: AMS, páginas brancas; DF, páginas amarelas, e RMA, gerenciador das próprias interfaces remotas da plataforma.

Figura 13 – Estrutura do projeto



Fonte: Próprio Autor.

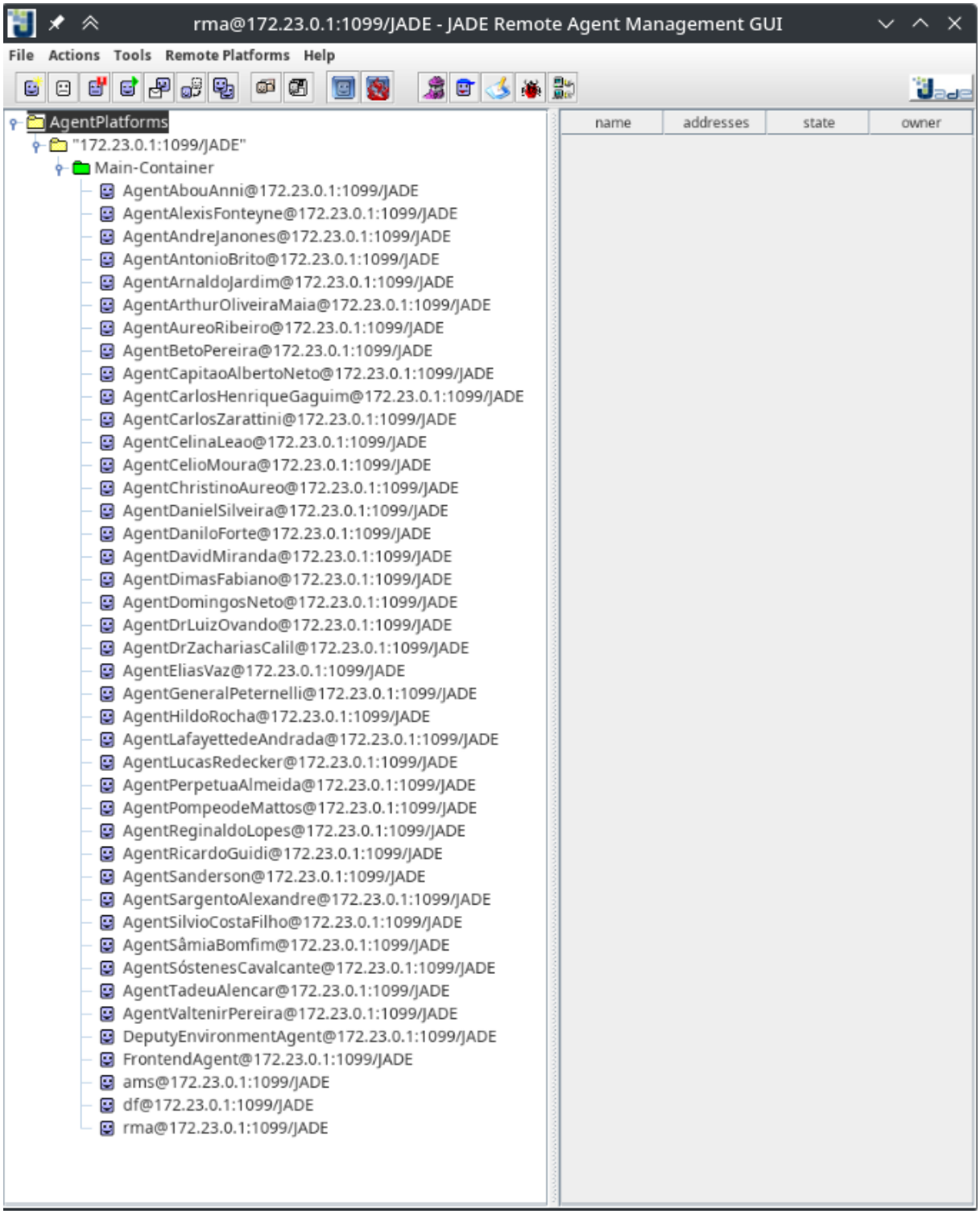
5.5 Considerações Finais do Capítulo

Este capítulo tratou das características da solução de software como um todo, abordando o contexto geral do tema; a base de dados que será utilizada no treinamento das LSTMs; as métricas (Acurácia, Precisão, *Recall*, *F1 Score* e Matriz de Confusão), e quais agentes foram planejados em uma visão preliminar da solução, encontrando-se em desenvolvimento pelo autor.

Foram abordadas, na contextualização, a crescente quantidade de dados legislativos, bem como a possibilidade de se trabalhar com eles em sistemas de simulação. Adicionalmente, mais detalhes sobre a base de dados utilizada para o treinamento foram acordados, conferindo uma visão arquitetural das LSTMs.

Por fim, foram apresentados os principais agentes que compõem a solução em desenvolvimento.

Figura 14 – Janela de gerenciamento de agentes do JADE



Fonte: Próprio Autor.

6 Análise de Resultados

Este capítulo apresenta a análise de resultados, com base nos insumos produzidos com a realização do projeto. Inicia-se com Ciclos de Teste, orientando-se pelo protocolo de pesquisa-ação, estabelecido em Método de Análise de Resultados. Esses ciclos de teste são apresentados via Ciclos de Teste, sendo tratados três cenários principais. Na sequência, constam Outras Iniciativas conferidas ao longo do trabalho. Por fim, são apresentadas as Considerações Finais do Autor.

6.1 Ciclos de Teste

Conforme citado na seção 4.4 Método de Análise de Resultados, esse trabalho orienta-se pelo método de Pesquisa-Ação. A pesquisa-ação segue um ciclo contínuo de planejamento, ação, observação e reflexão ao longo do processo. Especificamente, para o escopo desse trabalho, foi considerado um protocolo adaptado, contendo as etapas: definição do problema; pesquisa preliminar; desenvolvimento de um plano de ação; coleta de dados para avaliação dos efeitos da implementação do plano, e comunicação dos resultados. Entretanto, a definição do problema e a pesquisa preliminar já foram etapas reportadas em capítulos anteriores, nos quais se contextualiza e justifica a pertinência quanto à simulação de votação parlamentar usando Sistemas Multiagentes (Capítulo 1 e Capítulo 5, bem como são apresentados os embasamentos teóricos para uma possível solução ao problema contextualizado (Capítulo 2).

Diante do exposto, nesse momento de apresentação dos resultados, há detalhamentamento quanto às etapas: desenvolvimento de um plano de ação; coleta de dados para avaliação dos efeitos da implementação do plano, e comunicação dos resultados. No intuito de facilitar a compreensão dessas etapas, há: (i) um descritivo inicial sobre aspectos que são comumente encontrados em cada cenário estudado (ex. abordagem iterativa-incremental e métricas analisadas), e (ii) plano de ação, com a apresentação do cenário de teste. Posteriormente, são apresentados, de forma conjunta, a coleta de dados e os resultados obtidos.

6.1.1 Descritivo Inicial

Ressalta-se que foi conduzida uma abordagem iterativa-incremental ao conduzir cada cenário de treinamento do sistema de simulação. Portanto, em um primeiro momento, o treinamento foi realizado com poucos dados, os quais foram tendo amostras quantitativas de dados cada vez maiores, à medida que os cenários evoluem, tornando-os

mais complexos.

Em termos de métricas, foram analisadas em cada ciclo, ou seja, em cada cenário: Acurácia, Precisão, Revocação e Medida F1. Adicionalmente, para apresentar uma visão detalhada do desempenho do modelo treinado, fez-se uso de uma Matriz de Confusão, comparando previsões feitas pelo sistema de simulação de votação orientado a SMA com os resultados reais das votações. Essas métricas são apresentadas ao final do treinamento do modelo de um partido.

Há ainda a apresentação de algumas simulações realizadas na interface de usuário, permitindo comparar mais claramente as previsões dos agentes com os resultados reais de uma votação nominal.

6.1.2 Plano de Ação do Cenário de Teste 1

Neste primeiro cenário, foi avaliado o comportamento do sistema ao ser treinado com a entrada de poucos dados para treinamento e teste. Esses dados constam em arquivos *0.txt* e *1.txt*, dentro do diretório de cada partido, e representam as descrições das proposições daquele partido.

A quantidade de dados de entrada para esse primeiro teste pode ser visualizada no Quadro 8, o qual lista todos os dados dos partidos separados por treino e teste. O cenário é composto das seguintes parâmetros:

- Tratamento dos dados: Os arquivos foram previamente processados e nivelados para ficarem com o mesmo tamanho (prevalecendo o tamanho do menor), sendo esse um pré-requisito para o treinamento da LSTM proposta;

- Tamanho das frases: Cada arquivo possui frases com tamanhos variados, prevalecendo a descrição original da proposição, com a aplicação de um processo de normalização para remoção de caracteres especiais, objetivando manter somente letras e números;

- Modelo treinado: Ao final do treinamento, é possível entrar com a descrição de uma proposição, e o modelo retornará a classe que achar mais adequada. Para esse projeto, as classes possíveis são somente duas: *favor* e *against*. A primeira representada pelo número 1 (um), e a segunda pelo número 0 (zero), e

- Redes Neurais: Utilizou-se uma Rede Neural Recorrente em conjunto com a Memória de Curto e Longo Prazo.

Quadro 8 – Distribuição inicial de dados por partido

Nome do Partido	**Sigla**	**Treinamento**	**Teste**	**Total**
Avante	AVANTE	312	78	390
Cidadania	CIDADANIA	286	72	358
Movimento Democrático Brasileiro	MDB	701	176	877
Partido Novo	NOVO	394	99	493
Partido Comunista do Brasil	PCdoB	700	175	875
Partido Democrático Trabalhista	PDT	624	156	780
Partido Liberal	PL	452	114	566
Podemos	PODE	416	104	520
Progressistas	PP	806	202	1008
Partido Renovação Democrática*	PRD	-	-	-
Partido Socialista Brasileiro	PSB	638	160	798
Partido Social Democrático	PSD	824	207	1031
Partido da Social Democracia Brasileira	PSDB	704	176	880
Partido Socialismo e Liberdade	PSOL	656	165	821
Partido dos Trabalhadores	PT	941	236	1177
Partido Verde	PV	611	153	764
Rede Sustentabilidade	REDE	448	113	561
Republicanos	REPUBLICANOS	347	87	434
Solidariedade	SOLIDARIEDADE	462	116	578
União Brasil	UNIÃO	120	30	150

Fonte: Próprio Autor

* Não possui dados para treino

6.1.2.1 Coleta de Dados e Resultados Obtidos

Nesse primeiro teste, foi utilizada uma rede com duas camadas, sendo a primeira uma LSTM, e a segunda é uma camada de saída RNN Output para prever classes. Definiu-se ainda uma frequência mínima de duas palavras para que fosse considerada no modelo. Adicionalmente, estabeleceu-se uma única interação, indicando que o modelo passaria somente uma vez sobre o conjunto de dados. Como as palavras são convertidas em vetores, utilizando o *WordToVec*, foi definido um tamanho inicial de 100 dimensões para o vetor de palavras gerado. Por fim, o tamanho da janela de contexto, que captura as relações entre palavras, foi definido com o valor 20, ou seja, o modelo considera até 20 palavras

antes e depois da palavra atual ao calcular os *embeddings*.

O treinamento dos modelos utilizou somente uma época, e os registros de treinamento, número de interações, as épocas e a matriz de confusão podem ser visualizados na Wiki <https://douglasmonteles.github.io/politics-agents>. Acesso em: 13 fevereiro 2025.

Dos resultados, ao final do treinamento do modelo, uma instância de treino é executada com dados de teste que não foram utilizados no treinamento, e as métricas são automaticamente calculadas para cada modelo. Em resumo, o Quadro 9 apresenta as métricas obtidas, sendo que Precisão, Revocação e F1 foram calculadas apenas para a classe positiva, pois o *Deeplearning4j* considera essa a classe mais relevante para o problema de classificação.

Considerando a interface de usuário do sistema, a Figura 15 apresenta uma simulação, na qual os votos a favor foram previstos com uma taxa de acerto de quase 100%, atingindo 97,22%. Já na Figura 16, é possível visualizar o contrário, onde os votos contra foram previstos com 100% de acertos.

Analisando os resultados obtidos orientando-se pelas métricas e simulações, podem ser reportadas as seguintes observações:

- Baixa Acurácia Geral: A maioria dos partidos tem acurácia entre 0,44 e 0,52, o que indica que o modelo não está muito acima do acaso. Isso pode significar que o modelo não está conseguindo generalizar bem, evidenciando o desequilíbrio entre classes (alguns partidos possuem mais dados de treinamento do que outros);

- Baixa Precisão e Revocação para Alguns Partidos: Movimento Democrático Brasileiro (MDB), Progressistas (PP) e Partido Verde (PV) apresentam precisão e revocação extremamente baixas. Isso indica que o modelo treinado desses partidos tem muita dificuldade em identificar e classificar proposições;

- Muito Baixa Predição Observada para o Caso do Partido PSD: Partido Social Democrático (PSD) tem precisão, revocação e F1-score iguais a 0,00. Isso indica que o modelo nunca prevê corretamente as proposições desse partido;

- Muito Alta Revocação para Alguns Partidos (modelo tende a se comportar de forma empírica - ou seja, "modelo tende a chutar"): Partido Novo (0,96), PDT (0,91), PL (0,94), PSB (0,97), União Brasil (0,83), Republicanos (0,78). Revocação alta significa que quase todas as proposições reais desses partidos estão sendo capturadas, mas isso pode ser problemático, caso a precisão não seja alta;

- Melhor Equilíbrio Entre Precisão e Revocação Solidariedade (F1 = 0,60), União Brasil (F1 = 0,63), PDT (F1 = 0,67), PSB (F1 = 0,67): Esses partidos têm um equilíbrio

Quadro 9 – Métricas obtidas para cada modelo treinado

Nome do Partido	**Acurácia**	**Precisão**	**Revocação**	**F1**
Avante	0,50	0,51	0,26	0,35
Cidadania	0,44	0,43	0,36	0,39
Movimento Democrático Brasileiro	0,45	0,39	0,16	0,23
Partido Novo	0,50	0,50	0,96	0,66
Partido Comunista do Brasil	0,48	0,48	0,57	0,52
Partido Democrático Trabalhista	0,56	0,53	0,91	0,67
Partido Liberal	0,51	0,50	0,94	0,66
Podemos	0,46	0,44	0,26	0,33
Progressistas	0,45	0,38	0,16	0,23
Partido Renovação Democrática*	-	-	-	-
Partido Socialista Brasileiro	0,52	0,51	0,97	0,67
Partido Social Democrático	0,50	0,00	0,00	0,00
Partido da Social Democracia Brasileira	0,47	0,48	0,77	0,59
Partido Socialismo e Liberdade	0,50	0,50	0,52	0,51
Partido dos Trabalhadores	0,45	0,46	0,60	0,52
Partido Verde	0,49	0,42	0,03	0,07
Rede Sustentabilidade	0,51	0,52	0,45	0,48
Republicanos	0,50	0,50	0,78	0,61
Solidariedade	0,59	0,59	0,62	0,60
União Brasil	0,51	0,51	0,83	0,63

Fonte: Próprio Autor

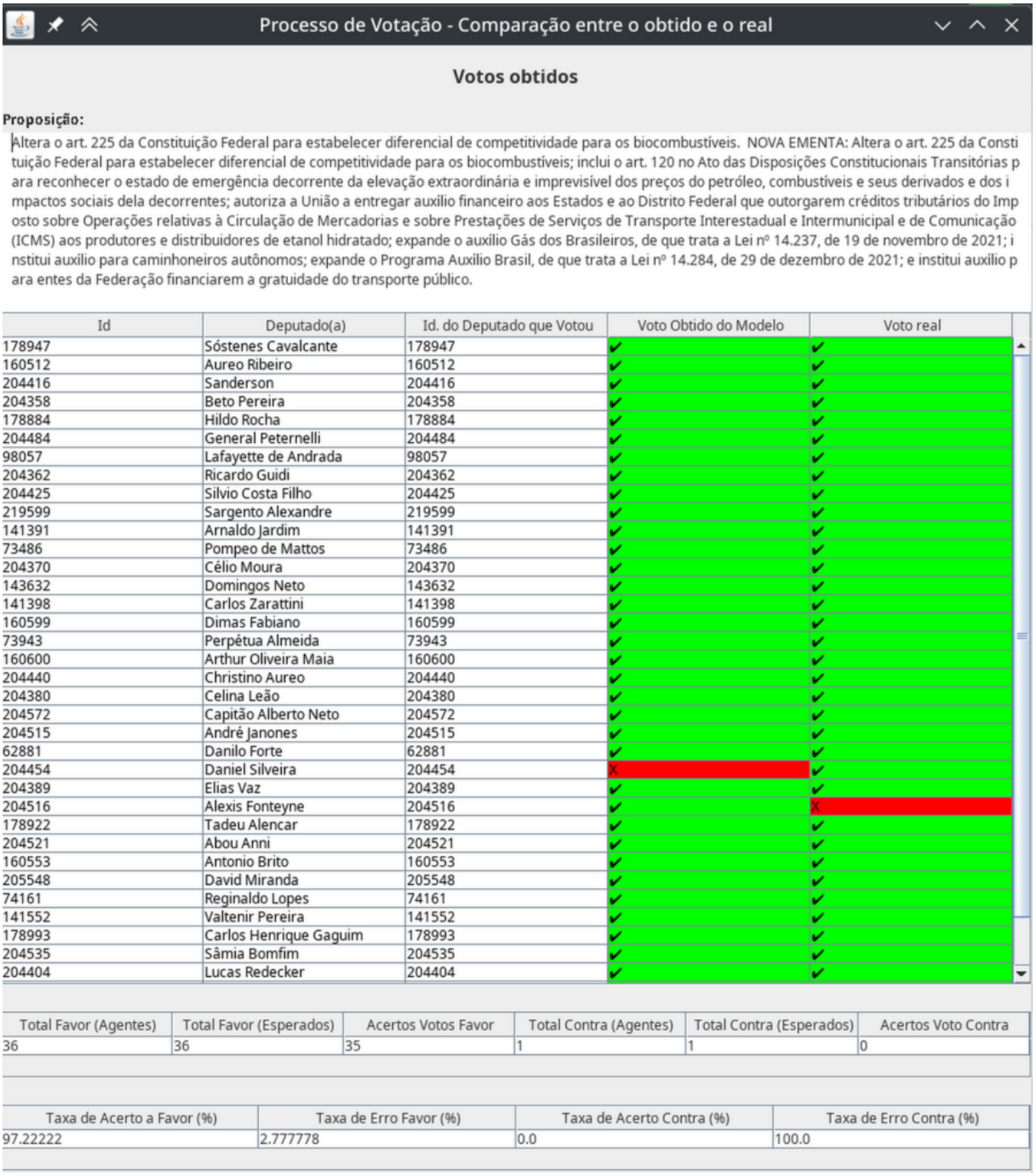
* Não possui dados para treino

melhor entre precisão e revocação, indicando que o modelo consegue identificá-los com mais confiança, e

- Maior Tendência de Votação dos Agentes (seja votando majoritariamente a favor, seja contra): Analisando as simulações, é possível perceber que os agentes, ao acessar o modelo treinado do respectivo partido, tendem a votar majoritariamente a favor ou contra, aumentando a taxa de acertos a favor, e praticamente zerando a taxa de acertos contra ou o contrário.

Dados os resultados do primeiro ciclo, para o Cenário de Teste 1, pouco satisfató-

Figura 15 – Simulação da votação com acertos nos resultados a favor

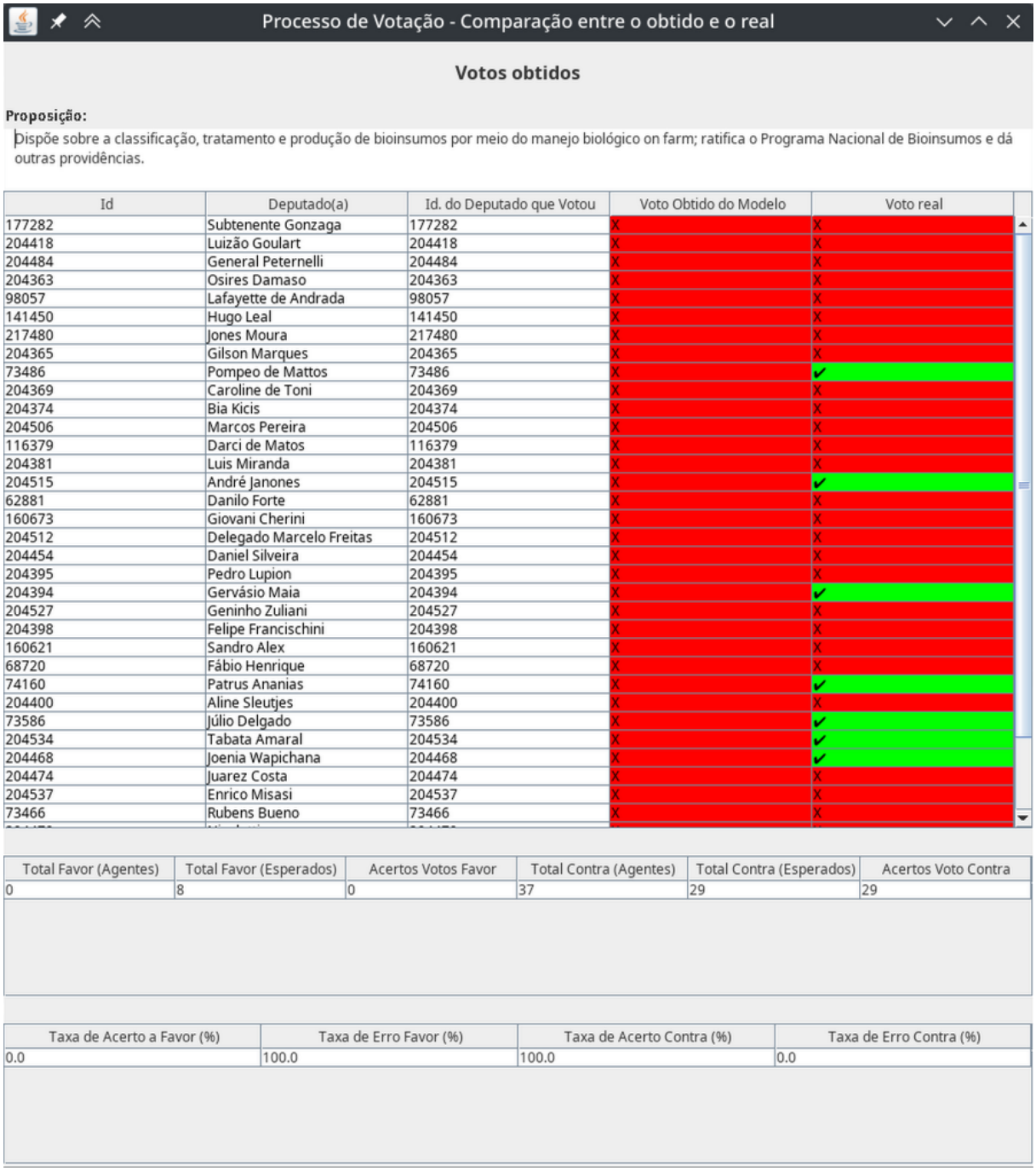


Fonte: Próprio Autor.

rios, um novo plano de ação foi estabelecido. A seguir, constam os principais elementos considerados:

- Aumentar dados de treino e teste: Os dados utilizados até o momento foram apenas as descrições das proposições. No próximo cenário de teste, foram adicionados dados obtidos dos PDFs das proposições, a fim de se obter mais informações relevantes e aumentar a base de dados;

Figura 16 – Simulação da votação com acertos nos resultados contra



Id	Deputado(a)	Id. do Deputado que Votou	Voto Obtido do Modelo	Voto real
177282	Subtenente Gonzaga	177282	X	X
204418	Luizão Goulart	204418	X	X
204484	General Peternelli	204484	X	X
204363	Osires Damaso	204363	X	X
98057	Lafayette de Andrada	98057	X	X
141450	Hugo Leal	141450	X	X
217480	Jones Moura	217480	X	X
204365	Gilson Marques	204365	X	X
73486	Pompeo de Mattos	73486	X	✔
204369	Caroline de Toni	204369	X	X
204374	Bia Kicis	204374	X	X
204506	Marcos Pereira	204506	X	X
116379	Darci de Matos	116379	X	X
204381	Luis Miranda	204381	X	X
204515	André Janones	204515	X	✔
62881	Danilo Forte	62881	X	X
160673	Giovani Cherini	160673	X	X
204512	Delegado Marcelo Freitas	204512	X	X
204454	Daniel Silveira	204454	X	X
204395	Pedro Lupion	204395	X	X
204394	Gervásio Maia	204394	X	✔
204527	Geninho Zuliani	204527	X	X
204398	Felipe Francischini	204398	X	X
160621	Sandro Alex	160621	X	X
68720	Fábio Henrique	68720	X	X
74160	Patrus Ananias	74160	X	✔
204400	Aline Sleutjes	204400	X	X
73586	Júlio Delgado	73586	X	✔
204534	Tabata Amaral	204534	X	✔
204468	Joenia Wapichana	204468	X	✔
204474	Juarez Costa	204474	X	X
204537	Enrico Misasi	204537	X	X
73466	Rubens Bueno	73466	X	X

Total Favor (Agentes)	Total Favor (Esperados)	Acertos Votos Favor	Total Contra (Agentes)	Total Contra (Esperados)	Acertos Voto Contra
0	8	0	37	29	29

Taxa de Acerto a Favor (%)	Taxa de Erro Favor (%)	Taxa de Acerto Contra (%)	Taxa de Erro Contra (%)
0.0	100.0	100.0	0.0

Fonte: Próprio Autor.

- Ajustar parâmetros de treino: Os parâmetros como número de frequência de uma palavra, número de interações, dimensões do vetor, tamanho da janela de contexto, e número de épocas serão ajustados, visando melhorar o resultado do treino, e

- Investigar o modelo do PSD: Com base no 9, é possível perceber que esse partido possui uma quantidade razoável de dados. Sendo assim, possivelmente, ocorreu algum erro nesse primeiro treinamento. Além disso, como foi apenas uma época, o modelo não conseguiu obter um resultado adequado.

6.1.3 Plano de Ação do Cenário de Teste 2

Neste segundo cenário, foram adicionados mais dados de treinamento e testes, no intuito de obter maior precisão nos resultados de classificação, bem como procurando mitigar os problemas encontrados no ciclo de testes anterior. Esse aumento na quantidade de dados de treino condiz com a abordagem iterativa-incremental adotada, e explicada no Descritivo Inicial, visando conferir maior complexidade ao modelo com o passar dos ciclos de teste.

Para esse cenário, os modelos foram treinados com as quantidades de dados apresentas no Quadro 10. O aumento da base de dados foi possível graças ao processo de conversão dos PDFs das proposições em texto, aumentando a quantidade de texto, e fornecendo informações detalhadas sobre as proposições.

Esse cenário está estruturado, em termos de tratamento de dados; tamanho das frases; modelo treinado, e redes neurais, da mesma forma que foi acordado para o caso do Cenário de Teste 1.

6.1.3.1 Coleta de Dados e Resultados Obtidos

Nesse segundo teste, foi utilizada uma rede com duas camadas, sendo a primeira uma LSTM, e a segunda é uma camada de saída RNN Output para prever classes. Adicionalmente, foi estabelecida uma frequência mínima de duas palavras para que fosse considerada no modelo. Definiu-se ainda o número de 50 interações, indicando que o modelo passaria 50 vezes sobre o conjunto de dados. Como as palavras são convertidas em vetores, utilizando o *WordToVec*, o tamanho inicial do vetor de palavras foi alterado para 200 dimensões. Por fim, o tamanho da janela de contexto, que captura as relações entre palavras, foi definido com o valor 40, ou seja, o modelo considera até 40 palavras antes e depois da palavra atual ao calcular os *embeddings*.

O treinamento dos modelos utilizou dez épocas, e os registros de treinamento, o número de interações, as épocas, e a matriz de confusão podem ser visualizados na Wiki <https://douglasmonteles.github.io/politics-agents>. Acesso em: 13 fevereiro 2025.

Dos resultados, ao final do treinamento do modelo, uma instância de treino foi executada com dados de teste que não foram utilizados no treinamento, e as métricas foram automaticamente calculadas para cada modelo. Em resumo, o Quadro 12 apresenta as métricas obtidas, sendo que Precisão, Revocação e F1 foram calculadas apenas para a classe positiva, pois o *Deeplearning4j* considera essa a classe mais relevante para o problema de classificação.

Os valores de acurácia, precisão, revocação e F1-score mostram como o modelo atuou - de forma mais preditiva - na classificação das proposições parlamentares por partido.

Quadro 10 – Distribuição de dados de treino e teste separados por partido

Nome do Partido	**Sigla**	**Treinamento**	**Teste**	**Total**
Avante	AVANTE	1909	478	2387
Cidadania	CIDADANIA	2169	543	2712
Movimento Democrático Brasileiro	MDB	5131	1283	6414
Partido Novo	NOVO	2898	725	3623
Partido Comunista do Brasil	PCdoB	5273	1319	6592
Partido Democrático Trabalhista	PDT	5229	1308	6537
Partido Liberal	PL	3587	897	4484
Podemos	PODE	2917	732	3659
Progressistas	PP	6066	1517	7583
Partido Renovação Democrática*	PRD	-	-	-
Partido Socialista Brasileiro	PSB	5149	1288	6437
Partido Social Democrático	PSD	6355	1589	7944
Partido da Social Democracia Brasileira	PSDB	4830	1280	6038
Partido Socialismo e Liberdade	PSOL	5263	1316	6579
Partido dos Trabalhadores	PT	7263	1816	9079
Partido Verde	PV	4273	1069	5342
Rede Sustentabilidade	REDE	3246	812	4058
Republicanos	REPUBLICANOS	2597	650	3247
Solidariedade	SOLIDARIEDADE	3058	765	3823
União Brasil	UNIÃO	912	228	1140

Fonte: Próprio Autor

* Não possui dados para treino

Analisando os resultados obtidos orientando-se pelas métricas simulações, podem ser reportadas as seguintes observações:

- Melhorias na acurácia, mas ainda próxima do acaso: A maioria dos partidos tem acurácia entre 0,46 e 0,57, o que revela uma leve melhora em relação aos resultados anteriores. No entanto, como um classificador aleatório esperaria atingir algo próximo de 50% em classes balanceadas, esse resultado ainda não é ideal;

- Melhorias na captura de algumas classes: O modelo melhorou na revocação para alguns partidos, como Movimento Democrático Brasileiro (MDB) com Revocação

Quadro 11 – Métricas obtidas para cada modelo treinado (Ao final da 10° época)

Nome do Partido	**Acurácia**	**Precisão**	**Revocação**	**F1**
Avante	0,55	0,55	0,56	0,55
Cidadania	0,50	0,50	0,43	0,46
Movimento Democrático Brasileiro	0,55	0,53	0,71	0,61
Partido Novo	0,46	0,46	0,39	0,42
Partido Comunista do Brasil	0,51	0,51	0,67	0,58
Partido Democrático Trabalhista	0,53	0,53	0,54	0,54
Partido Liberal	0,52	0,52	0,51	0,51
Podemos	0,53	0,53	0,56	0,54
Progressistas	0,54	0,52	0,80	0,63
Partido Renovação Democrática*	-	-	-	-
Partido Socialista Brasileiro	0,50	0,50	0,56	0,53
Partido Social Democrático	0,57	0,59	0,49	0,53
Partido da Social Democracia Brasileira	0,52	0,52	0,63	0,57
Partido Socialismo e Liberdade	0,46	0,47	0,51	0,49
Partido dos Trabalhadores	0,55	0,54	0,59	0,56
Partido Verde	0,49	0,49	0,46	0,47
Rede Sustentabilidade	0,50	0,50	0,51	0,51
Republicanos	0,52	0,52	0,51	0,52
Solidariedade	0,57	0,57	0,56	0,57
União Brasil	0,48	0,49	0,93	0,64

Fonte: Próprio Autor

* Não possui dados para treino

0,71 (antes 0,16), Progressistas (PP) com Revocação 0,80 (antes 0,16), União Brasil com Revocação 0,93 (antes 0,83), e Partido Comunista do Brasil (PCdoB) com Revocação 0,67 (antes 0,57). Isso significa que o modelo está conseguindo identificar melhor à qual classe pertence a proposição para esses partidos;

- Imprecisões Recorrentes: Alguns partidos continuam com precisão abaixo de 0,50, como Partido Novo (0,46), PSOL (0,47), Partido Verde (0,49) e União Brasil (0,49). Isso significa que quando o modelo prevê a classe de voto de um desses partidos, ele frequentemente está errado, e

- Melhorias nos acertos: Para votos a favor e contra, a taxa de acerto cresceu em relação ao teste anterior. Entretanto, com novos ajustes de parâmetros do treino, talvez seja possível melhorar o resultado ainda mais, uma vez que o mesmo está abaixo de 50%.

Considerando uma simulação na interface de usuário, Figura 17, percebe-se, com base na quantidade de acertos de votos a favor e contra que os agentes acertaram mais de 90% dos votos a favor. Em relação aos votos contra, a taxa de acerto foi superior a 40%, indicando que os agentes estão conseguindo votar com maior equilíbrio.

Dados os resultados do segundo ciclo, para o Cenário de Teste 2, pouco satisfatórios na taxa de acerto de ambas as classes, um novo plano de ação foi estabelecido. A seguir, constam os principais elementos considerados:

- Analisar o balanceamento dos dados: Certificar-se de que cada partido tem um número razoável de exemplos;
- Aprimorar *embeddings*: Melhorar a representação textual das proposições, ajustando os parâmetros do *WordToVec*, e
- Testar ajustes na rede LSTM: Ajustar o valor dos parâmetros de treinamento e aumentar o número de épocas do treinamento do modelo.

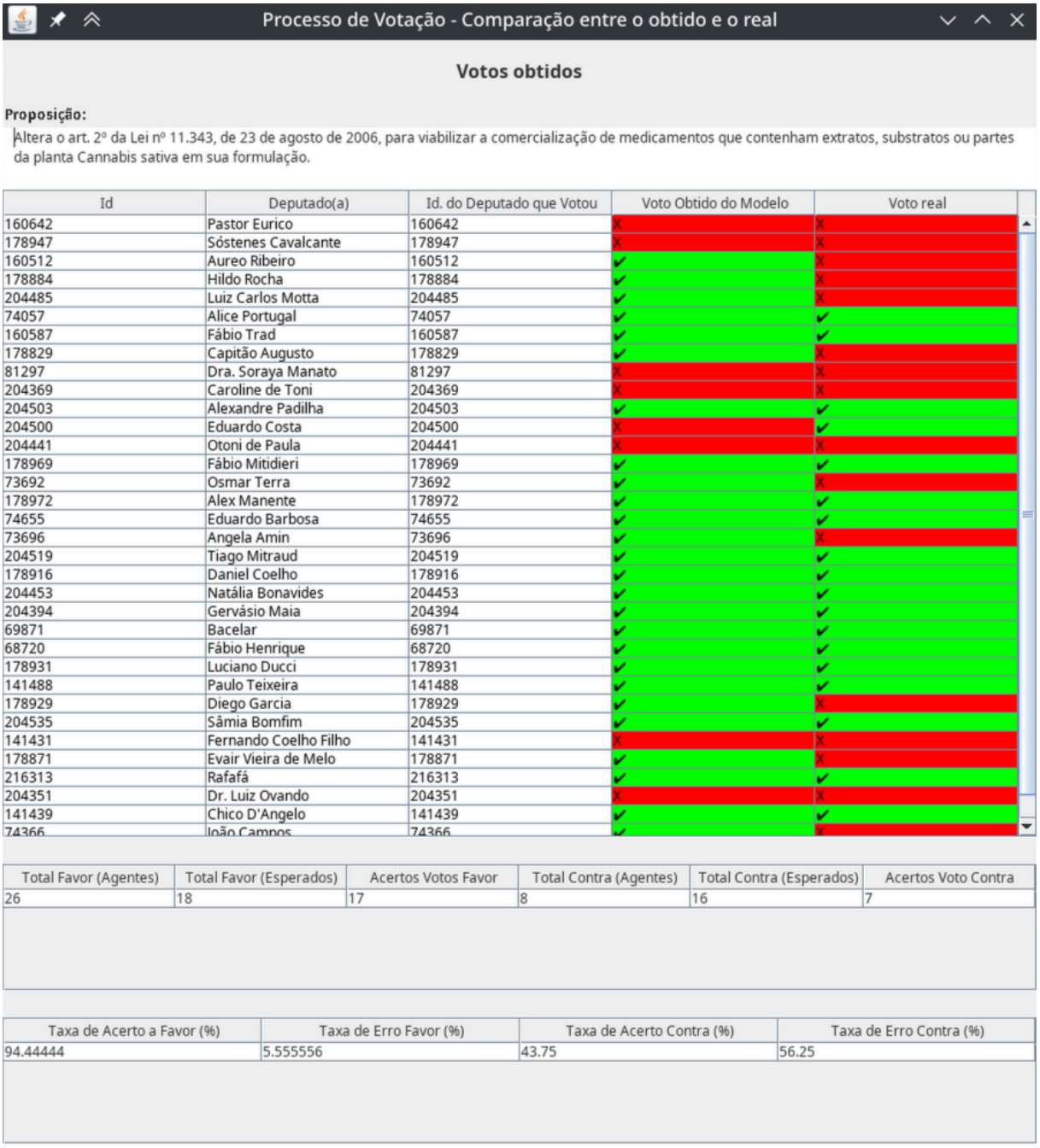
6.1.4 Plano de Ação do Cenário de Teste 3

Para o terceiro caso de teste, os modelos foram treinados com a mesma quantidade de dados do teste anterior (vide Quadro 10). Objetiva-se melhorar os resultados de treinamento alterando alguns parâmetros no processo de conversão de palavras em vetores com o *Word2Vec* e no treinamento dos modelos.

Esse cenário está estruturado, em termos de tratamento de dados; tamanho das frases; modelo treinado, e redes neurais, da mesma forma que foi acordado para o caso do Cenário de Teste 1. Entretanto, constam algumas alterações de parâmetros para os casos:

- Alteração dos parâmetros de treinamento do *Word2Vec*: Aumento no número de interações realizadas pelo *Word2Vec*, melhorando a qualidade dos *embeddings* de palavras. Aumento no número de dimensões do vetor de palavras e no tamanho da janela de contexto, e
- Alteração dos parâmetros de treinamento do modelo: Aumento do tamanho do *batchSize*, visando maior eficiência e melhor consumo de hardware. Aumentou-se ainda o número de épocas para 80, visando permitir um melhor ajuste dos pesos e dada a pouca quantidade de dados para treino.

Figura 17 – Simulação de votação com equilíbrio de resultados



Processo de Votação - Comparação entre o obtido e o real

Votos obtidos

Proposição:

Altera o art. 2º da Lei nº 11.343, de 23 de agosto de 2006, para viabilizar a comercialização de medicamentos que contenham extratos, substratos ou partes da planta Cannabis sativa em sua formulação.

Id	Deputado(a)	Id. do Deputado que Votou	Voto Obtido do Modelo	Voto real
160642	Pastor Eurico	160642	X	X
178947	Sóstenes Cavalcante	178947	X	X
160512	Aureo Ribeiro	160512	✔	X
178884	Hildo Rocha	178884	✔	X
204485	Luiz Carlos Motta	204485	✔	X
74057	Alice Portugal	74057	✔	✔
160587	Fábio Trad	160587	✔	✔
178829	Capitão Augusto	178829	✔	X
81297	Dra. Soraya Manato	81297	X	X
204369	Caroline de Toni	204369	X	X
204503	Alexandre Padilha	204503	✔	✔
204500	Eduardo Costa	204500	X	✔
204441	Otoni de Paula	204441	X	X
178969	Fábio Mitidieri	178969	✔	✔
73692	Osmar Terra	73692	✔	X
178972	Alex Manente	178972	✔	✔
74655	Eduardo Barbosa	74655	✔	✔
73696	Angela Amin	73696	✔	X
204519	Tiago Mitraud	204519	✔	✔
178916	Daniel Coelho	178916	✔	✔
204453	Natália Bonavides	204453	✔	✔
204394	Gervásio Maia	204394	✔	✔
69871	Bacelar	69871	✔	✔
68720	Fábio Henrique	68720	✔	✔
178931	Luciano Ducci	178931	✔	✔
141488	Paulo Teixeira	141488	✔	✔
178929	Diego Garcia	178929	✔	X
204535	Sâmia Bomfim	204535	✔	✔
141431	Fernando Coelho Filho	141431	X	X
178871	Evair Vieira de Melo	178871	✔	X
216313	Rafafá	216313	✔	✔
204351	Dr. Luiz Ovando	204351	X	X
141439	Chico D'Angelo	141439	✔	✔
74366	João Campos	74366	✔	X

Total Favor (Agentes)	Total Favor (Esperados)	Acertos Votos Favor	Total Contra (Agentes)	Total Contra (Esperados)	Acertos Voto Contra
26	18	17	8	16	7

Taxa de Acerto a Favor (%)	Taxa de Erro Favor (%)	Taxa de Acerto Contra (%)	Taxa de Erro Contra (%)
94.44444	5.555556	43.75	56.25

Fonte: Próprio Autor.

6.1.4.1 Coleta de Dados e Resultados Obtidos

Nesse terceiro teste, foi utilizada uma rede com duas camadas, sendo a primeira uma LSTM, e a segunda é uma camada de saída RNN Output para prever classes. Estabeleceu-se uma frequência mínima de duas palavras para que fosse considerada no modelo. Definiu-se ainda o número de 200 interações, indicando que o modelo passaria 200 vezes sobre o conjunto de dados. Como as palavras foram convertidas em vetores, utilizando o *WordToVec*, o tamanho inicial do vetor de palavras foi alterado para 400 dimensões. Por fim, o tamanho da janela de contexto, que captura as relações entre pa-

lavras, foi definido com o valor 60, ou seja, o modelo considera até 20 palavras antes e depois da palavra atual ao calcular os embeddings e o *batchSize* para 256.

O treinamento dos modelos utilizou oitenta épocas, e os registros de treinamento, o número de interações, as épocas, e a matriz de confusão podem ser visualizados na Wiki <https://douglasmonteles.github.io/politics-agents>. Acesso em: 13 fevereiro 2025.

Dos resultados, ao final do treinamento do modelo, uma instância de treino foi executada com dados de teste que não foram utilizados no treinamento, e as métricas foram são automaticamente calculadas para cada modelo. Em resumo, o Quadro 12 apresenta as métricas obtidas, sendo que Precisão, Revocação e F1 foram calculadas apenas para a classe positiva, pois o *Deeplearning4j* considera essa a classe mais relevante para o problema de classificação.

Considerando algumas simulações de votação na tela para visualizar o comportamento dos agentes e os votos sobre um determinado tema, percebe-se, Figura 18, que tanto a taxa de acertos de votos a favor; quanto a taxa de acertos de votos contra estão relativamente altas. Além disso, na Figura 19, é possível notar que ainda existem casos em que há discrepância entre as taxas de acerto de votos a favor e contra.

6.2 Outras Iniciativas

Ocorreram, ao longo da realização desse trabalho, outras iniciativas teórico-práticas, que demandaram esforço de pesquisa, programação, configuração de ambiente, planejamento, tomada de decisão, dentre outros. Nesse contexto, cabe mencionar sobre:

- Redes Neurais: foram analisadas várias redes neurais adicionais, as quais não foram efetivamente utilizadas no trabalho, uma vez que não atendiam as especificidades do projeto. Essa análise consta documentada no Apêndice G, com foco em Redes Adversárias Generativa; Redes de Hopfield e Máquinas de Boltzmann; Máquinas de Estado Líquido e Máquinas de Estado de Eco; Máquinas de Turing Neurais e Computadores Neurais Diferenciáveis; Redes Residuais Profundas; Redes de Kohonen, e Redes de Cápsulas;

- Fluxos de Processamento: foram modelados, em notação formal BPMN, cada fluxo de processamento, considerando a descrição das proposições; o treinamento do modelo baseado na descrição das proposições; e a interação entre os agentes deputados. Essas modelagens constam, respectivamente, nas Figuras 5, 8 e 11, e

- Condução do Projeto: foram conduzidas as atividades de desenvolvimento do projeto, conforme planejado no Método de Desenvolvimento. Evidências sobre *Backlog* do Produto; *Backlog* das Principais *Sprints*; Testes de Desempenho por *Sprint*;

Quadro 12 – Métricas obtidas para cada modelo treinado (Ao final da 80° época)

Nome do Partido	**Acurácia**	**Precisão**	**Revocação**	**F1**
Avante	0,53	0,53	0,52	0,53
Cidadania	0,48	0,48	0,51	0,50
Movimento Democrático Brasileiro	0,58	0,56	0,73	0,63
Partido Novo	0,36	0,38	0,46	0,42
Partido Comunista do Brasil	0,50	0,50	0,51	0,50
Partido Democrático Trabalhista	0,50	0,50	0,61	0,55
Partido Liberal	0,49	0,49	0,47	0,48
Podemos	0,48	0,47	0,33	0,39
Progressistas	0,52	0,52	0,55	0,54
Partido Renovação Democrática*	-	-	-	-
Partido Socialista Brasileiro	0,46	0,45	0,33	0,38
Partido Social Democrático	0,53	0,53	0,52	0,52
Partido da Social Democracia Brasileira	0,57	0,57	0,60	0,58
Partido Socialismo e Liberdade	0,45	0,46	0,50	0,48
Partido dos Trabalhadores	0,52	0,52	0,47	0,50
Partido Verde	0,51	0,51	0,66	0,57
Rede Sustentabilidade	0,54	0,56	0,42	0,48
Republicanos	0,51	0,51	0,52	0,52
Solidariedade	0,53	0,52	0,62	0,57
União Brasil	0,46	0,45	0,36	0,40

Fonte: Próprio Autor

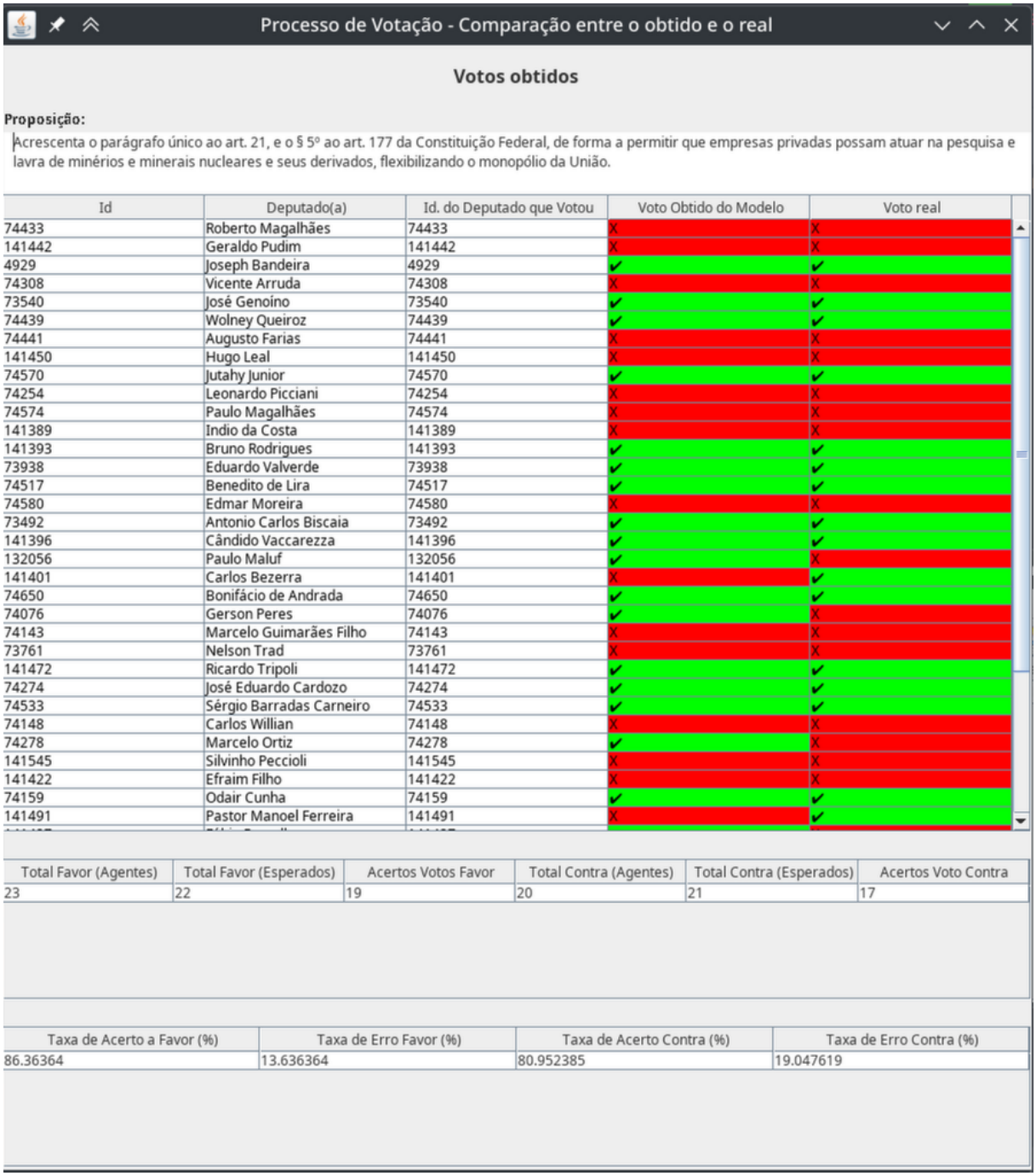
* Não possui dados para treino

Retrospectiva Planejamento das *Sprints*, e Quadro *Kanban* podem ser consultadas no repositório do projeto, mais especificamente na Wiki disponível em: <https://douglasmonteles.github.io/politics-agents>. Acesso em: 13 fevereiro 2025.

6.3 Considerações Finais do Capítulo

Neste capítulo, foram apresentados alguns testes realizados ao final do treinamento dos modelos para cada partido, com diferentes quantidades de dados e de parâmetros. Foram utilizadas métricas de Acurácia, Precisão, Revocação e F1. Em cada teste, foram

Figura 18 – Simulação da votação acertos de voto a favor e contra



Id	Deputado(a)	Id. do Deputado que Votou	Voto Obtido do Modelo	Voto real
74433	Roberto Magalhães	74433	X	X
141442	Geraldo Pudim	141442	X	X
4929	Joseph Bandeira	4929	✔	✔
74308	Vicente Arruda	74308	X	X
73540	José Genoíno	73540	✔	✔
74439	Wolney Queiroz	74439	✔	✔
74441	Augusto Farias	74441	X	X
141450	Hugo Leal	141450	X	X
74570	Jutahy Junior	74570	✔	✔
74254	Leonardo Picciani	74254	X	X
74574	Paulo Magalhães	74574	X	X
141389	Indio da Costa	141389	X	X
141393	Bruno Rodrigues	141393	✔	✔
73938	Eduardo Valverde	73938	✔	✔
74517	Benedito de Lira	74517	✔	✔
74580	Edmar Moreira	74580	X	X
73492	Antonio Carlos Biscaia	73492	✔	✔
141396	Cândido Vaccarezza	141396	✔	✔
132056	Paulo Maluf	132056	✔	X
141401	Carlos Bezerra	141401	X	✔
74650	Bonifácio de Andrada	74650	✔	✔
74076	Gerson Peres	74076	✔	X
74143	Marcelo Guimarães Filho	74143	X	X
73761	Nelson Trad	73761	X	X
141472	Ricardo Tripoli	141472	✔	✔
74274	José Eduardo Cardozo	74274	✔	✔
74533	Sérgio Barradas Carneiro	74533	✔	✔
74148	Carlos Willian	74148	X	X
74278	Marcelo Ortiz	74278	✔	X
141545	Silvinho Peccioli	141545	X	X
141422	Efraim Filho	141422	X	X
74159	Odair Cunha	74159	✔	✔
141491	Pastor Manoel Ferreira	141491	X	✔

Total Favor (Agentes)	Total Favor (Esperados)	Acertos Votos Favor	Total Contra (Agentes)	Total Contra (Esperados)	Acertos Voto Contra
23	22	19	20	21	17

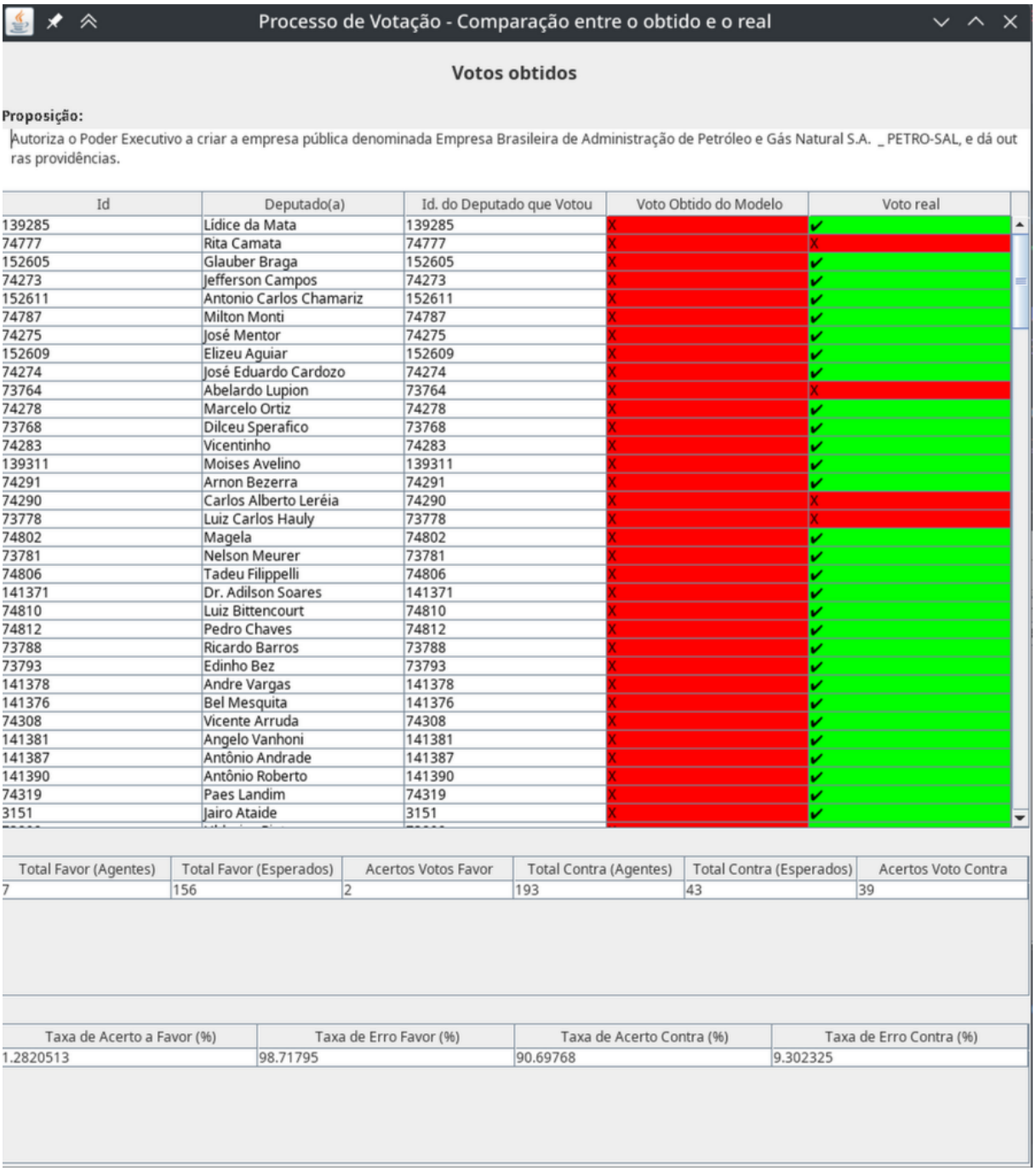
Taxa de Acerto a Favor (%)	Taxa de Erro Favor (%)	Taxa de Acerto Contra (%)	Taxa de Erro Contra (%)
86.36364	13.636364	80.952385	19.047619

Fonte: Próprio Autor.

obtidos resultados que permitiram a evolução do próximo ciclo.

Em cada um dos testes, foram treinados modelos para cada partido, recebendo como dados de treinamento/teste as descrições das proposições, cujas orientações do partido era voto a favor ou contra. Todos os testes apresentaram evoluções quanto à predição dos votos, sendo o terceiro o que mais se destacou durante as simulações, apresentando ótimas previsões, tanto de votos a favor; quanto contra, na mesma votação.

Figura 19 – Simulação da votação com erros de voto a favor



Fonte: Próprio Autor.

7 Conclusão

Esse capítulo tem por finalidade apresentar as considerações finais em relação à pesquisa que foi desenvolvida. Inicialmente, será abordado o Contexto Geral do projeto, que apresenta as justificativas da pesquisa. Em seguida, é apresentado o Estado Atual do Trabalho, que elenca os objetivos que foram cumpridos e responde à questão de pesquisa. Na sequência, é destaca-se às Contribuições e Limitações desse projeto. Por fim, dado que esse projeto possui potenciais evoluções, são apresentados os Trabalhos Futuros e as Considerações Finais do Autor.

7.1 Contexto Geral

A digitalização do processo legislativo gerou um grande volume de dados sobre proposições parlamentares e votações, possibilitando a aplicação de técnicas de aprendizado de máquina para prever resultados de deliberações. No entanto, devido à complexidade das interações políticas e à influência de diversos fatores externos, a previsão de votações continua sendo um desafio. Diante disso, este trabalho atuou na combinação de Redes Neurais Recorrentes do tipo *Long Short-Term Memory* (LSTM) e Sistemas Multiagentes (SMA) com a plataforma JADE para modelar e simular o comportamento dos deputados, no intuito de predizer sobre os resultados de votações futuras.

A abordagem adotada envolve a criação de um modelo de rede neural treinado para cada partido, permitindo capturar padrões específicos de comportamento nas votações. O modelo é alimentado com a descrição textual das proposições parlamentares, separadas em duas classificações: 1 (favor) ou 0 (contra). A escolha da LSTM justifica-se pela sua capacidade de processar sequências textuais e capturar dependências temporais, sendo ideal para análise do impacto legislativo ao longo do tempo.

Além disso, o trabalho implementa um Sistema Multiagentes (SMA), no qual cada deputado é representado por um agente individual dentro da plataforma JADE. Esses agentes simulam interações políticas e consultam o modelo do respectivo partido para tomar decisões sobre novas proposições. Dessa forma, a simulação permite analisar diferentes cenários de votação, considerando fatores relacionados às tendências históricas.

Como propósito, esse trabalho centralizou esforços na obtenção de um modelo preditivo capaz de auxiliar na compreensão das tendências partidárias, facilitando a análise de possíveis resultados antes mesmo da realização de uma votação. Além disso, a pesquisa contribui para o desenvolvimento de futuras ferramentas que podem ser utilizadas tanto por analistas políticos; quanto por cidadãos interessados no processo legislativo.

7.2 Estado Atual do Trabalho

O Objetivo Geral, definido para esse trabalho, compreende: "Desenvolvimento de um sistema multiagentes capaz de deliberar, tomar uma decisão com base em informações disponíveis, sobre as proposições discutidas na Câmara dos Deputados e apresentar uma simulação da votação com base em temas votados anteriormente e nas características partidárias individuais de cada deputado.". Esse objetivo foi cumprido. O sistema multiagentes criado simula votações com agentes atuando como deputados(as), e votando a favor ou contra uma dada proposição. Comprobatórios quanto ao atendimento do Objetivo Geral encontram-se, principalmente, documentados no Capitulo 5.

Quanto aos Objetivos Específicos, podem ser mencionadas as seguintes considerações:

- Estudo sobre a base de dados que contém as informações sobre as proposições discutidas na Câmara dos Deputados, os resultados das votações e as informações relacionadas aos deputados, como as orientações de voto dos seus respectivos partidos.
  - Status: Cumprido.
  - Comprobatórios: Capítulo 5, seção 5.2 e Apêndices do A ao C.
- Estudo sobre o paradigma de Sistemas Multiagentes.
  - Status: Cumprido.
  - Comprobatórios: Capítulo 2, seção 2.1.
- Estudo sobre processamento de dados e algoritmos associados (ex. Aprendizado de Máquina e Aprendizado Profundo com Redes Neurais Artificiais).
  - Status: Cumprido.
  - Comprobatórios: Capítulo 2, seções 2.2, 2.3 e Capítulo 5.
- Identificação de quais categorias referenciadas nas bases foram de fato consumidas no processamento de dados: ora para o levantamento das proposições discutidas na Câmara dos Deputados, ora para deliberar sobre os assuntos e estabelecer a votação.
  - Status: Cumprido.
  - Comprobatórios: Capítulo 5, seção 5.2 e 5.3.2.
- Modelagem do Sistema Multiagentes capaz de processar os dados inerentes ao contexto e deliberar sobre.
  - Status: Cumprido.

  - Comprobatórios: Capítulo 5, seção 5.4.

- Disponibilização do Sistema Multiagentes para conhecimento dos interessados, e.
  - Status: Cumprido.
  - Comprobatórios: Capítulo 5, seção 5.1.

- Análise do Sistema Multiagentes, considerando diferentes configurações de entrada.
  - Status: Cumprido.
  - Comprobatórios: Capítulo 6, seção 6.1.

Estabeleceu-se ainda uma Questão de Pesquisa, a qual norteou o trabalho como um todo, sendo a mesma: "É possível criar um Sistema Multiagentes que consiga deliberar sobre as proposições apresentadas à Câmara dos Deputados, simulando o processo de votação com base em resultados anteriores sobre temas semelhantes?". Como resposta ao questionamento, tem-se que a mesma é positiva. Foi possível criar um sistema multiagentes com esse propósito, e a solução foi desenvolvida e disponibilizada para testes.

Apesar dos resultados obtidos serem relevantes, indicando a pertinência do projeto, há como melhorar o processo de predição. Em especial, podem ser refinados ainda mais os modelos treinados, evoluindo-os em vários aspectos (ex. uso de mais épocas, aumento no quantitativo de dados, dentre outros), até que os mesmos permitam obter resultados cada vez mais satisfatórios.

7.3 Contribuições e Limitações

O sistema criado inova ao combinar redes neurais recorrentes (LSTM) com sistemas multiagentes (SMA), utilizando a biblioteca *Deeplearning4j*, e permitindo uma análise mais rica e dinâmica do cenário legislativo. Essa abordagem simula diferentes cenários políticos; testa hipóteses sobre mudanças no comportamento partidário, e até identifica padrões emergentes na deliberação legislativa, algo que não seria possível apenas com aprendizado de máquina ou apenas com simulações baseadas em regras.

Ressalta-se ainda, como contribuições desse trabalho, a própria disponibilização de uma base de dados, com dados separados por partidos, bem como com as proposições categorizadas em "a favor"e "contra". Essa base pode ajudar trabalhos futuros que possuem alguma correlação com o presente trabalho, ou ainda para aqueles que desejam usar os dados da Câmara dos Deputados relacionados às proposições partidárias. A base completa encontra-se disponível em: <https://github.com/DouglasMonteles/politics-agents-external-files>.

Destacam-se ainda outras contribuições desse projeto, e que conferem insumos relevantes em vários âmbitos, seja de pesquisa, seja de desenvolvimento, seja de análise de resultados. Seguem algumas para conhecimento:

- Levantamento de referencial embasatório sobre Sistemas Multiagentes (Capítulo 2 - Referencial Teórico - seção 2.1);
- Levantamento sobre diferentes redes neurais (Apêndices G);
- Ambiente integrado para desenvolvimento, combinando redes neurais e SMA, e orientando-se por tecnologias da comunidade Java (<https://github.com/DouglasMonteles/politics-agents/blob/main/Makefile>).
- Modelagens de fluxos, contextualizando cada processo de treinamento com especificação de processo em notação formal BPMN, para os casos: Fluxo de processamento dos dados para treinamento, Fluxo de treinamento dos modelos de cada partido e Fluxo de comunicação entre os agentes, e
- Identificação de um conjunto de métricas que sejam pertinentes para esse perfil de projeto, com destaque para Acurácia, Precisão, Revocação e F1, além da Matriz de Confusão (Capítulo 5, seção 5.3.3 Métricas).

Todo trabalho, mesmo que contendo resultados muito favoráveis, possuem limitações. Como limitações desse trabalho, podem ser ressaltadas:

- Há necessidade de aumentar a quantidade de informações de alguns partidos. Isso ocorre, pois alguns partidos não tiveram seus dados utilizados no treinamento, uma vez que esses dados - quando analisados - não atendiam aos critérios de processamento estabelecidos. Nesse sentido, o simulador não atuou de forma tão precisa - em termos de predições - para esses casos;
- Há necessidade de considerar - adicionalmente - dados de partidos mais antigos. A base de dados da Câmara dos Deputados fornece uma lista com os partidos. Essa lista foi utilizada para separar cada um dos partidos. Porém, somente os partidos que estão atualmente ativos são retornados nessa lista. Nesse contexto, partidos mais antigos, e que não existem mais, não foram considerados. Entretanto, isso faz com que o sistema desconsidere um conjunto relevante de votações passadas;
- Há necessidade de aumentar a quantidade de dados no treinamento. Como as proposições tratam de temas diversos, inerentes à sociedade, deve-se ter à disposição uma grande quantidade de dados, visando modelos cada vez mais bem treinados. Mesmo obtendo dados das proposições via PDFs, o que já representa um quantitativo de

dados razoável (97.678), ainda é considerada baixa essa quantidade de dados para viabilizar um bom processo de treinamento;

- Há necessidade de observar as particularidades das conversões, preparando funções que tratem esses casos, para que o treinamento do modelo não tenha que lidar com muitos "ruídos". Como a conversão de PDF para texto puro não é uma tarefa simples, pode acontecer da conversão resultar em palavras quebradas ou caracteres especiais, que não puderam ser convertidos, ou seja, "ruídos";
- Há necessidade de extrapolar um pouco mais a proposta, visando generalizar a solução para textos de proposições que não receberam treinamento, e
- Há necessidade de aumentar a quantidade de dados, uma vez que o modelo ainda encontra dificuldades em predizer corretamente o voto "a favor"ou "contra", para o caso de algumas proposições.

7.4 Trabalhos Futuros

Como trabalhos futuros, podem ser tratadas cada uma das necessidades acordadas nas limitações. Adicionalmente, outras sugestões encontram-se apresentadas na sequência, atuando com foco em outras abordagens exploratórias:

- Incorporar novas fontes de dados, como debates parlamentares, menções em redes sociais e notícias, para refinar as predições;
- Criar um modelo específico para cada deputado, considerando suas preferências individuais, em vez de treinar um modelo por partido;
- Utilizar análise de sentimentos em discursos parlamentares para enriquecer a tomada de decisão dos agentes. LLMs podem ser aplicados para identificar emoções; perceber o tom linguístico de um conteúdo, e até mesmo analisar sentimentos, tal como abordado em (IMPROTA, 2023) e (MOREIRA; AL., 2024). No primeiro caso, o autor Alexandre Improta incorpora análise de sentimentos na predição de séries temporais financeiras. Já no segundo caso, os autores Moreira et al. (2024) fazem uma comparação de ferramentas de análise de sentimentos aplicadas no contexto educacional;
- Criar agentes que representem outros atores políticos (*lobby*, imprensa, opinião pública) e suas influências nas decisões parlamentares, e
- Aplicar uma análise temporal sobre as orientações de votos dos partidos, levando em consideração as mudanças de opinião ao longos dos governos e verificando se há uma mudança no padrão de votação para um mesmo tema.

7.5 Considerações Finais do Autor

Para o autor, foi muito interessante desenvolver esse trabalho, com o qual várias lições foram aprendidas:

- Aplicação de Redes Neurais Recorrentes (LSTM) para modelagem e previsão de votações parlamentares;
- Implementação de Sistemas Multiagentes (SMA) com JADE, simulando uma pequena parte do processo de deliberação política;
- Integração de IA com Modelagem Política, avaliando sua aplicabilidade e limitações;
- A orientação partidária tem grande peso na decisão do voto dos deputados;
- A deliberação legislativa é complexa e influenciada por fatores externos, além dos dados históricos;
- Prever votações parlamentares é desafiador devido à imprevisibilidade política, e
- Testes e ajustes contínuos são necessários para otimizar modelos de IA.

Referências

AHMAD, M. O.; MARKKULA, J.; OIVO, M. Kanban in software development: A systematic literature review. In: *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. Santander: IEEE, 2013. p. 9–16. Citado na página 69.

Apache Maven. *What is Maven?* Acesso em: 15 jun. 2024. Disponível em: <https://maven.apache.org/what-is-maven.html>. Citado na página 47.

Apache Software Foundation. *OpenNLP - Apache Documentation*. 2024. <https://opennlp.apache.org/docs/2.3.3/manual/opennlp.html#opennlp>. Acesso em: 19 mai. 2024. Citado 2 vezes nas páginas 50 e 51.

BALDASSIN, A. J.; GUILHERME, I. R.; MALTEMPI, M. V. Uma abordagem baseada em agentes para filtragem de correspondências eletrônicas. *Revista Eletrônica de Iniciação Científica (REIC)*, v. 2, n. 4, 2002. Citado 3 vezes nas páginas 32, 33 e 56.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, Springer, p. 353–374, 2023. Citado na página 38.

BARBOSA, M. E.; MAAS, H.; PEREIRA, B. H. Análise de sentimento por meio de redes neurais artificiais em sistemas de avaliação de cursos de graduação. 2019. Citado na página 35.

BELGHACHE, E.; GEORGÉ, J.-P.; GLEIZES, M.-P. Towards an adaptive multi-agent system for dynamic big data analytics. In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. [S.l.: s.n.], 2016. p. 753–758. Citado 2 vezes nas páginas 25 e 68.

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. Developing multi-agent systems with jade. In: SPRINGER. *Intelligent Agents VII Agent Theories Architectures and Languages: 7th International Workshop, ATAL 2000 Boston, MA, USA, July 7–9, 2000 Proceedings 7*. [S.l.], 2001. p. 89–103. Citado na página 28.

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. JADE - a JAVA agent development framework. *Multi-Agent Systems and Applications: Proceedings of the Second International Symposium*, p. 47–53, 2001. Citado na página 41.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, v. 5, n. 2, p. 157–166, 1994. Citado 3 vezes nas páginas 39, 68 e 81.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2006. (Information Science and Statistics). Citado na página 37.

BROWN, A. W. *An Annotated Bibliography on Integration in Software Engineering Environments*. Pittsburgh, Pennsylvania, 1992. Citado na página 45.

CASTOLDI, A. C.; SANTOS, M. d. O. d. *Raciocínio baseado em casos*. [S.l.]: Florianópolis, 2002. Citado 2 vezes nas páginas 32 e 33.

Câmara dos Deputados. *Dados Abertos da Câmara dos Deputados*. 2024. Acesso em: 20 mar. 2024. Disponível em: <https://dadosabertos.camara.leg.br/swagger/api.html#api>. Citado 4 vezes nas páginas 25, 78, 147 e 148.

DS4SD Team. *Docling: Documentation for Linguistics*. 2025. [Accessed January 20, 2025]. Disponível em: <https://ds4sd.github.io/docling/>. Citado na página 56.

ELMAN, J. Finding structure in time. *Cognitive Science*, v. 14, p. 179–211, 1990. Citado na página 38.

ENGEL, G. I. Pesquisa-ação. *Educar em Revista*, SciELO Brasil, p. 181–191, 2000. Citado na página 69.

FERREIRA, A. B. H. *Novo Dicionário Aurélio - Século XXI*. 3. ed. [S.l.]: Editora Nova Fronteira, 1999. Citado na página 33.

FONSECA, J. J. S. *Metodologia da pesquisa científica*. Fortaleza: UEC, 2002. Apostila. Citado 4 vezes nas páginas 63, 64, 65 e 66.

FONSECA, J. M. M. R. *Protocolos de Negociação com Coligações em Sistemas Multi-agente*. Tese (Doutorado) — Universidade Nova de Lisboa, 2001. Citado na página 33.

FRANCELIN, R. A.; VIEIRA, L. R. Rede de kohonen. 1994. Citado na página 141.

FUADDAH, N. S.; MAHARANI, M. M. Microsoft teams in the perspective of the users. *Journal of Advanced Multidisciplinary Research*, v. 2, n. 2, p. 62–69, 2021. Citado na página 58.

GEEKSFORGEEKS. *Echo State Network - An Overview*. Acesso em: 15 jun. 2024. <https://www.geeksforgeeks.org/echo-state-network-an-overview/>. Citado na página 139.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa*. [S.l.]: Plageder, 2009. Citado na página 63.

GIL, A. C. *Como elaborar projetos de pesquisa*. 4. ed. São Paulo: Atlas, 2007. Citado 2 vezes nas páginas 64 e 65.

GitHub. *About Git*. Acesso em: 15 jun. 2024. Disponível em: <https://docs.github.com/pt/get-started/using-git/about-git>. Citado 2 vezes nas páginas 45 e 46.

GOLDENBERG, M. *A arte de pesquisar*. Rio de Janeiro: Record, 1997. Citado na página 64.

GOODFELLOW, I. et al. Generative adversarial nets. In: *Proceedings of the Advances in Neural Information Processing Systems*. Montreal, QC, Canada: [s.n.], 2014. Citado na página 137.

GRAVES, A.; WAYNE, G.; DANIHELKE, I. Neural turing machines. *arXiv*, arXiv:1410.5401, 2014. Citado na página 140.

GRAVES, A. et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, v. 538, p. 471–476, 2016. Citado na página 140.

HAYKIN, S. Redes neurais- princípios e práticas. *BOOKMAN, São Paulo*, v. 2, p. 900, 2001. Citado na página 35.

HE, K. et al. Deep residual learning for image recognition. *arXiv*, arXiv:1512.03385, 2015. Citado na página 140.

HOSSAIN, E.; BABAR, M. A.; PAIK, H. Using scrum in global software development: A systematic literature review. In: *Fourth IEEE International Conference on Global Software Engineering*. Limerick: [s.n.], 2009. p. 175–184. Citado na página 68.

HOUDT, G. V.; MOSQUERA, C.; NÁPOLES, G. A review on the long short-term memory model. *Artificial Intelligence Review*, Springer, v. 53, n. 8, p. 5929–5955, 2020. Citado 2 vezes nas páginas 39 e 40.

HUANG, G.; ZHU, Q.; SIEW, C. Extreme learning machine: Theory and applications. *Neurocomputing*, v. 70, p. 489–501, 2006. Citado na página 139.

HÜBNER, J. F.; SICHMAN, J. S. Organização de sistemas multiagentes. *bolsa*, v. 680033, p. 99–8, 2003. Citado na página 31.

IBM. *How Java Works*. Acesso em: 15 jun. 2024. Disponível em: <https://www.ibm.com/topics/java#How+Java+works>. Citado na página 46.

IBM. *IBM Developer Kit for Linux, Java Edition Documentation*. Acesso em: 15 jun. 2024. Disponível em: <https://www.ibm.com/docs/en/i/7.4?topic=platform-java-development-kit>. Citado na página 47.

IBM. *IBM Java Virtual Machine Documentation*. Acesso em 15 jun. 2024. Disponível em: <https://www.ibm.com/docs/en/i/7.4?topic=platform-java-virtual-machine>. Citado na página 47.

IMPROTA, A. *Título do Trabalho (se disponível)*. 2023. [Acessado em: 11 fev. 2025]. Disponível em: <https://bdta.abcd.usp.br/directbitstream/aa074227-cb33-4cff-a6de-099f775784a0/Alexandre%20Improta.pdf>. Citado na página 113.

JADERBERG, M.; SIMONYAN, K.; ZISSERMAN, A. Spatial transformer network. In: *Proceedings of the Advances in Neural Information Processing Systems*. Montreal, QC, Canada: [s.n.], 2015. p. 2017–2025. Citado na página 38.

JAEGER, H.; HAAS, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, v. 304, p. 78–80, 2004. Citado na página 138.

JetBrains. *IntelliJ IDEA Download*. Acesso em: 15 jun. 2024. Disponível em: <https://www.jetbrains.com/idea/download/?section=linux>. Citado na página 46.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd (draft). ed. Prentice Hall, 2019. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>. Citado 3 vezes nas páginas 34, 84 e 85.

KAUR, A. App review: Trello. *Journal of Hospital Librarianship*, Taylor & Francis, v. 18, n. 1, p. 95–101, 2018. Citado 2 vezes nas páginas 57 e 58.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. Disponível em: <http://arxiv.org/abs/1312.6114>. Citado na página 37.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, v. 43, p. 59–69, 1982. Citado na página 141.

Konduit AI. *DeepLearning4j*. 2024. <https://deeplearning4j.konduit.ai/>. Acesso em: 19 mai. 2024. Citado na página 54.

KONDUIT AI. *DeepLearning4j - Language Processing Tutorials*. 2024. <https://deeplearning4j.konduit.ai/deeplearning4j/tutorials/language-processing>. Acesso em: 19 maio 2024. Citado 3 vezes nas páginas 54, 55 e 56.

Konduit AI. *Word2Vec - DeepLearning4j*. 2024. <https://deeplearning4j.konduit.ai/v/en-1.0.0-beta7/language-processing/word2vec>. Acesso em: 19 mai. 2024. Citado 3 vezes nas páginas 51, 52 e 53.

KULKARNI, T. et al. Deep convolutional inverse graphics network. In: *Proceedings of the Advances in Neural Information Processing Systems*. Montreal, QC, Canada: [s.n.], 2015. Citado na página 36.

KUMAR, E. *Natural Language Processing*. [S.l.]: I. K. International Pvt Ltd, 2011. Citado na página 34.

KUMAR, S.; KUMAR, U. Java agent development framework. *International Journal Research*, Citeseer, v. 1, n. 9, p. 1022–1025, 2014. Citado na página 41.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, p. 2278–2324, 1998. Citado na página 36.

LEIJNEN, S.; VEEN, F. v. The neural network zoo. In: MDPI. *Proceedings*. [S.l.], 2020. v. 47, n. 1, p. 9. Citado 11 vezes nas páginas 36, 37, 38, 39, 137, 138, 139, 140, 141, 142 e 143.

LIU, B. *Synthesis Lectures on Human Language Technologies*. [S.l.]: Morgan & Claypool, 2012. Citado na página 34.

LOBATO, A. G. P. et al. Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos. *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC*, 2016. Citado na página 24.

MAAS, W.; NATSCHLäGER, T.; MARKRAM, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, v. 14, p. 2531–2560, 2002. Citado na página 138.

MALIKA, F. Knowledge management and information technology. In: *2014 4th International Symposium ISKO-Maghreb: Concepts and Tools for knowledge Management (ISKO-Maghreb)*. [S.l.: s.n.], 2014. p. 1–7. Citado 2 vezes nas páginas 23 e 24.

MOREIRA, N.; AL. et. Título do Artigo (se disponível). *Anais do STIL*, 2024. [Acessado em: 11 fev. 2025]. Disponível em: <https://sol.sbc.org.br/index.php/stil/article/download/31165/30968/>. Citado na página 113.

NASHOLD, L.; KRISHNAN, R. Using lstm and sarima models to forecast cluster cpu usage. *arXiv preprint arXiv:2007.08092*, 2020. Citado na página 39.

NIKRAZ, M.; CAIRE, G.; BAHRI, P. A. A methodology for the analysis and design of multi-agent systems using jade. *International Journal of Computer Systems Science and Engineering*, v. 21, n. 2, p. 1–40, 2006. Citado na página 49.

NURSEITOV, N. et al. Comparison of json and xml data interchange formats: a case study. *Caine*, Citeseer, v. 9, p. 157–162, 2009. Citado na página 25.

OLAH, C. *Understanding LSTM networks*. 2015. Citado na página 39.

OLIVEIRA, L. F. de. Transmissão sináptica. *Brazilian Journal of Anesthesiology*, Sociedade Brasileira de Anestesiologia (SBA), v. 44, n. 1, p. 25–33, 2020. Citado na página 142.

OLIVEIRA, R. N. d. et al. Aprendizagem de máquina em um ambiente para negociações automatizadas. Universidade Federal de Campina Grande, 2006. Citado na página 34.

OSÓRIO, F. S.; BITTENCOURT, J. R.; OSÓRIO, F. S. Sistemas inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens. In: SN. *I Workshop de inteligência artificial*. [S.l.], 2000. Citado na página 35.

OVERLEAF. *Learn LaTeX in 30 minutes*. 2024. Acesso em: 22 mai. 2024. Disponível em: <https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes>. Citado na página 60.

OVERLEAF. *Overleaf*. 2024. Acesso em: 22 mai. 2024. Disponível em: <https://www.overleaf.com/about>. Citado na página 60.

PACHECO, L. B. *Como se fazem as leis*. [S.l.]: Edições Câmara, 2021. Citado na página 28.

PONCE, L. M. S. Extensao de um ambiente de computaçao de alto desempenho para o processamento de dados massivos. Universidade Federal de Minas Gerais, 2018. Citado na página 25.

RAMOS, A. Guia rápido: Gcc, makefile e valgrind. 2015. Citado na página 56.

RODRIGUES, F. A. *DSpace - RIUFCG: Repositório Institucional da Universidade Federal de Campina Grande*. 2002. Acesso em: 28 mai. 2024. Disponível em: <http://dspace.sti.ufcg.edu.br:8080/xmlui/handle/riufcg/4314>. Citado na página 38.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, p. 386, 1958. Citado 2 vezes nas páginas 35 e 50.

SABOUR, S.; FROSST, N.; HINTON, G. Dynamic routing between capsules. In: *Proceedings of the Advances in Neural Information Processing Systems*. Long Beach, CA, USA: [s.n.], 2017. p. 3856–3866. Citado na página 142.

SHARMA, D.; SHADABI, F. Multi-agents based data mining for intelligent decision support systems. In: *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*. [S.l.: s.n.], 2014. p. 241–245. Citado 3 vezes nas páginas 25, 28 e 68.

SHOHAM, Y. Agent oriented programming. *Artificial Intelligence*, v. 60, n. 1, p. 51–92, 1993. Citado na página 48.

SILVA, L. *Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework*. Fortaleza: [s.n.], 2002. Citado na página 43.

SILVEIRA, R. A. Introdução a sistemas multiagente. *Universidade Federal de Santa Catarina (UFSC)*, 2006. Citado 2 vezes nas páginas 31 e 86.

SITUMORANG, A. S. Microsoft teams for education sebagai media pembelajaran. *Journal of Mathematics Education and Applied*, v. 02, n. 01, p. 30–35, 2020. Citado na página 58.

SPöRL, C.; CASTRO, E. G.; LUCHIARI, A. Aplicação de redes neurais artificiais na construção de modelos de fragilidade ambiental. *Revista do Departamento de Geografia*, v. 21, n. 1, p. 113–135, 2011. Citado na página 35.

Techtudo. *Astah Community - Tudo sobre Astah Community*. 2025. [Acessado em: 11 fev. 2025]. Disponível em: <https://www.techtudo.com.br/tudo-sobre/astah-commmunity/>. Citado na página 57.

TECKCHANDANI, A. *Slack: A unified communications platform to improve team collaboration*. [S.l.]: Academy of Management Briarcliff Manor, NY, 2018. Citado 2 vezes nas páginas 58 e 59.

TEIXEIRA, F. V. *JADE: Java Agent Development Framework*. 2010. Available online. Disponível em: <http://www.dca.fee.unicamp.br/~gudwin/courses/IA>. Citado 3 vezes nas páginas 41, 42 e 43.

Telecom Italia Lab. *JADE - Java Agent Development Framework*. Acesso em: 15 jun. 2024. Disponível em: <https://jade.tilab.com/>. Citado na página 48.

Telegram Messenger LLP. *Telegram FAQ*. Acesso em: 15 jun. 2024. Disponível em: <https://telegram.org/faq#p-o-que-e-telegram-o-que-faco-aqui>. Citado 2 vezes nas páginas 59 e 60.

The Asimov Institute. *The Neural Network Zoo*. Acesso em: 10 de abr. 2024. Disponível em: <http://www.asimovinstitute.org/neural-network-zoo>. Citado 2 vezes nas páginas 35 e 67.

TRIVIñOS, A. N. S. *Introdução à pesquisa em ciências sociais: a pesquisa qualitativa em educação*. São Paulo: Atlas, 1987. Citado na página 65.

Visual Paradigm. *Visual Paradigm Online: Ferramenta de Modelagem e Design*. 2025. Acessado em: 13 fev. 2025. Disponível em: <https://online.visual-paradigm.com/pt/>. Citado na página 57.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, v. 10, n. 2, p. 115–152, 1995. Citado na página 48.

YUAN-YUAN, D.; YONG-CHENG, W.; HONG-MEI, W. The design and implement of e-government information system based on knowledge management. In: *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. [S.l.: s.n.], 2011. p. 2131–2134. Citado 2 vezes nas páginas 23 e 67.

ZEILER, M. et al. Deconvolutional networks. In: *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. San Francisco, CA, USA: [s.n.], 2010. Citado na página 36.

Apêndices

APÊNDICE A – Estrutura dos dados dos Deputados

```
Deputado {
   email: string
   id: integer($int64)
   idLegislatura: integer($int64)
   nome: string
   siglaPartido: string
   siglaUf: string
   uri: string
   uriPartido: string
   urlFoto: string
}
```

APÊNDICE B – Estrutura dos dados dos Partidos

```
Partido {
    id: string;
    nome: string;
    sigla: string;
    uri:  string;
}
```

APÊNDICE C – Estrutura dos dados do Voto

```
Voto {
    dataRegistroVoto: string;
    deputado_: Deputado;
    tipoVoto: string;
}
```

APÊNDICE D – Estrutura dos dados das Votações

```
Votacao {
   aprovacao: integer($int64)
   data:  string
   dataHoraRegistro: string
   descricao: string
   id:  string
   proposicaoObjeto: string
   siglaOrgao:  string
   uri: string
   uriEvento: string
   uriOrgao: string
   uriProposicaoObjeto: string
}
```

APÊNDICE E – Estrutura dos dados das Proposições

```
ProposicaoGeral {
   ano: integer($int64)
   apreciacao:  Apreciacao {
      descricao: string
      id: integer($int64)
   }
   autor: Autor {
      codPartido:  integer($int64)
      ideCadastro: integer($int64)
      nome:  string
      siglaPartido: string
      siglaUF:  string
   }
   dataApresentacao: string
   ementa: string
   explicacaoEmenta: string
   id:  integer($int64)
   indGenero: string
   nome:  string
   numero: integer($int64)
   orgaoNumerador: OrgaoNumerador{
   id:  integer($int64)
   nome:   string
   sigla: string
   qtdAutores:  integer($int64)
   qtdOrgaosComEstado: integer($int64)
   regime: Regime {
      descricao: string
      id: integer($int64)
   }
   situacao: Situacao {
      descricao: string
      id:  integer($int64)
      orgao: OrgaoSituacao {
         codOrgaoEstado: integer($int64)
```

```
        siglaOrgaoEstado: string
      }
      principal: Principal {
        codProposicaoPrincipal: integer($int64)
        proposicaoPrincipal: string
      }
    }
    tipo:  TipoProposicao {
      id:  integer($int64)
      nome:  string
      sigla: string
    }
    ultimoDespacho: UltimoDespacho{
    data:  string($date-time)
    descricao: string
  }
```

APÊNDICE F – Estrutura dos dados das Orientações de Voto fornecidas pelos Partidos

```
Orientacao{
    codPartidoBloco: integer($int64)
    codTipoLideranca: string
    orientacaoVoto: string
    siglaPartidoBloco: string
    uriPartidoBloco: string
}
```

APÊNDICE G – Redes Neurais que foram analisadas, mas que não seriam úteis para o presente trabalho

G.1 Redes Adversárias Generativa

As Redes Adversárias Generativas (GANs), de acordo com Goodfellow et al. (2014), consistem em duas redes: uma encarregada de gerar dados (o gerador), e outra de prever se os dados foram gerados ou não (o discriminador). O sucesso preditivo do discriminador é usado como um gradiente de erro para o gerador. Esse arranjo visa fazer com que o discriminador se torne melhor em distinguir dados reais de dados gerados, enquanto o gerador aprende a se tornar menos previsível.

Esse jogo dinâmico pode ser visto como uma espécie de teste de *Turing*, ou um correlato neural do algoritmo *Minimax*. Esse algoritmo é muito usado em Teoria de Jogos, principalmente quando, no jogo, se um jogador ganha, o outro necessariamente perde. Imaginando que, ao final do jogo, os cenários possíveis são: Ganhar, Empatar ou Perder. Nesse caso, bastaria a IA buscar por caminhos que levam ao ganho, que a vitória estará garantida (LEIJNEN; VEEN, 2020).

No algoritmo *Minimax*, deve-se mapear os caminhos possíveis em uma estrutura de dados do tipo árvore; depois, percorrer os nós dessa árvore até chegar nos nós de término de jogo, identificando se o jogador ganhou, empatou ou perdeu. Normalmente, conferem-se valores aos possíveis resultados. No caso, seriam: 1 para Ganhar, 0 para Empatar, e -1 para Perder. Ao visitar o nó em um nível anterior, é possível identificar de quem é a vez (jogador ou oponente). Se oponente, o algoritmo guarda o Min (menor resultado dos caminhos investigados). Se jogador, o algoritmo guarda o Max (maior resultado dos caminhos investigados). Repete-se esse processo, até o nó raiz da árvore. Percebe-se que, caso a decisão do jogador ganhar ou perder esteja a critério do oponente, o algoritmo recomenda não ser pertinente aquele caminho, e indica os caminhos nos quais o jogador tem chance de ganhar a partida. Isso ocorre mesmo considerando boas jogadas do oponente (LEIJNEN; VEEN, 2020).

Em redes adversárias generativas, o processo de aprendizado é relativamente difícil de equilibrar, pois não irá convergir quando o gerador ou o discriminador for muito bem-sucedido em sua respectiva tarefa (LEIJNEN; VEEN, 2020).

G.2 Redes de *Hopfield* e Máquinas de *Boltzmann*

Nas redes de *Hopfield*, cada neurônio está conectado a todos os outros neurônios. Todos os neurônios são tanto nós de entrada quanto de saída (LEIJNEN; VEEN, 2020).

Máquinas de *Boltzmann* (Restritas) são semelhantes na medida em que apenas alguns neurônios são neurônios de entrada, enquanto outros são ocultos. Máquinas de *Boltzmann* restritas não possuem conectividade total entre neurônios, tornando-as tipicamente mais eficientes para serem usadas no aprendizado, particularmente quando empilhadas umas sobre as outras em uma uma rede conhecida como "crença profunda" (LEIJNEN; VEEN, 2020).

Redes de *Hopfield* e Máquinas de *Boltzmann* são treinadas fixando o valor dos neurônios de entrada ao padrão desejado, após o qual os pesos são aprendidos. Uma vez treinada, a rede irá convergir para um dos padrões aprendidos e permanecer estável em um desses estados de convergência, em parte devido à energia total na rede ser reduzida incrementalmente durante o treinamento, similar ao modelo de *Ising* (LEIJNEN; VEEN, 2020).

Esses tipos de redes também são chamados de memórias associativas porque convergem para o estado mais similar em comparação à sua entrada. Cadeias de *Markov*, embora não sejam arquiteturas de redes neurais, também são incluídas nesta visão geral, pois podem ser consideradas como predecessoras (LEIJNEN; VEEN, 2020).

G.3 Máquinas de Estado Líquido e Máquinas de Estado de Eco

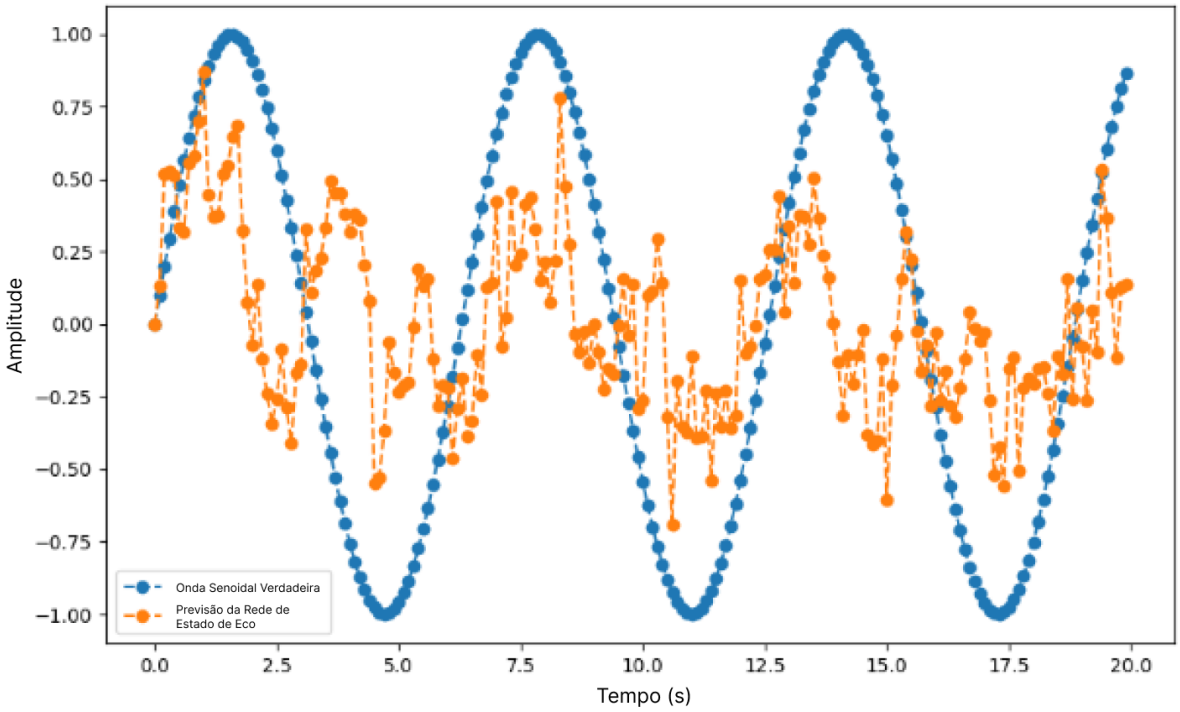
As Máquinas de Estado Líquido (do inglês, *Liquid State Machines* - LSMs), segundo Maas, Natschläger e Markram (2002), não são organizadas em camadas ordenadas, mas sim com conexões desenhadas aleatoriamente entre neurônios com funções de limiar, as quais permitem a acumulação de atividade ao longo do tempo, criando padrões de atividade pulsante. Consequentemente, em vez de usar a retropropagação, os neurônios de entrada são ativados e os sinais de atividade são propagados para frente através dos neurônios ocultos. Por isso, atividade pulsante. A propagação resultante dos sinais é usada para aprendizado por uma rede observadora separada que produz a saída.

As Máquinas de Estado de Eco (do inglês, *Echo State Machines* - ESMs), de acordo com Jaeger e Haas (2004), substituem esses neurônios pulsantes por neurônios de ativação sigmoide regulares. Conhecida também como função logística, a função sigmoide produz valores no intervalo [0, 1]. Isso a torna não-linear. Portanto, considera-se sua maior vantagem o fato de que o valor da derivada é máximo quando x está próximo de 0. Em resumo, com isso, tende-se a "impulsionar" o resultado para as extremidades do intervalo [0, 1] ao longo do treinamento, sendo algo desejável, principalmente, em problemas de

classificação binária. Esses problemas podem ser compreendidos via a probabilidade de algo pertencer ou não à determinada classe. Em contrapartida, a não-linearidade impõe altos custos computacionais, além de não ser uma boa opção para ativação das camadas intermediárias.

À exemplo, na Figura 20, uma ESN é utilizada com dados sintéticos gerados por programação, onde um vetor de tempo com passo de 0,1 no intervalo de 0 a 20 cria uma onda senoidal, à qual é adicionado ruído aleatório. Com um reservatório de tamanho 50, a ESN é treinada para prever padrões da onda senoidal a partir dos dados ruidosos, utilizando pares de entrada (ruído) e alvo (onda senoidal) para aprender a dinâmica subjacente. Os dados de treinamento são espelhados para criar dados de teste, assegurando um formato consistente para avaliação de desempenho. As previsões feitas pela ESN são comparadas visualmente à onda senoidal genuína (GEEKSFORGEEKS, Acesso em: 15 jun. 2024).

Figura 20 – Rede de estado de eco aprendendo a gerar onda senoidal



Fonte: GeeksforGeeks (Acesso em: 15 jun. 2024).

As Máquinas de Aprendizado Extremo (do inglês, *Extreme Learning Machines* - ELMs), segundo Huang, Zhu e Siew (2006), são semelhantes, mas não têm conexões recorrentes, permitindo que sejam treinadas rapidamente usando um algoritmo de aprendizado baseado em ajuste de mínimos quadrados (LEIJNEN; VEEN, 2020). Esse algoritmo é um método de otimização para estimar parâmetros desconhecidos em um modelo de regressão linear. Com isso, é possível encontrar o melhor ajuste para um conjunto de dados, procurando minimizar a soma dos quadrados das diferenças entre o valor estimado e os

dados reais observados. A essas diferenças dá-se o nome de resíduos. Interessante aplicá-lo quando a variância dos erros não é a mesma; ou quando há certa correlação entre os resíduos.

G.4 Máquinas de Turing Neurais e Computadores Neurais Diferenciáveis

As Máquinas de Turing Neurais (do inglês, *Neural Turing Machines* - NTMs), segundo Graves, Wayne e Danihelke (2014), podem ser entendidas como uma abstração das LSTMs, e uma tentativa de tornar as redes neurais mais explicáveis. Em vez de codificar uma célula de memória em um neurônio, a memória é separada como uma memória endereçável por conteúdo, onde a rede neural pode escrever e ler, tornando-as completas em termos de *Turing*.

Os Computadores Neurais Diferenciáveis (do inglês, *Differentiable Neural Computers* - DNCs), segundo Graves et al. (2016), são uma abstração adicional, com memórias escaláveis. Eles também apresentam três mecanismos de atenção que permitem à rede consultar a similaridade da entrada com as entradas de memória; a relação temporal entre duas entradas de memória, e se uma entrada de memória foi recentemente atualizada (LEIJNEN; VEEN, 2020).

Essas redes são particularmente interessantes pois, como mostrado por Graves et al. (2016), possibilitam alcançar um melhor desempenho em várias tarefas algorítmicas, incluindo o "recall associativo", que se assemelha à modelagem de sequência, e consiste em pedir ao modelo para recordar um item de uma lista consultando-o com o item precedente.

G.5 Redes Residuais Profundas

Redes Residuais Profundas (do inglês, *Deep Residual Networks* - ResNets), são outro exemplo de arquitetura de rede que não possui camadas estruturadas HE et al. (2015). *ResNets* são redes de propagação, nas quais as conexões podem atravessar qualquer número de camadas ocultas. Isso as torna semelhantes às redes neurais recorrentes, mas sem a estrutura de preservação do tempo (LEIJNEN; VEEN, 2020).

Diante do exposto, trata-se de uma rede capaz de lidar com modelos de aprendizagem profunda mais sofisticados. Nesses casos, um modelo de aprendizagem residual auxilia na preservação de bons resultados em uma rede com muitas camadas. Normalmente, em redes com múltiplas camadas, a precisão tende a não ser mantida ao longo do processo de aprendizagem. Blocos residuais preservam as entradas, e possibilitam o uso em treinamentos mais específicos, conforme consta acordado pelo autor Kaiming He, em

um tutorial intitulado “ICML 2016 Tutorial on Deep Residual Networks” [1].

G.6 Redes de *Kohonen*

As Redes de *Kohonen*, segundo Kohonen (1982), ou mapas auto-organizáveis, utilizam o aprendizado competitivo para classificar dados de entrada sem conhecer a saída esperada, utilizando uma função objetivo estética para classificação bem-sucedida. Essa função objetivo estética pode ser entendia como um conjunto de regras que permite aprender e calcular um mapeamento de entrada-saída com propriedades desejáveis específicas (LEIJNEN; VEEN, 2020).

O principal objetivo de um algoritmo de aprendizagem auto-organizada é descobrir padrões significativos ou características nos dados de entrada. Após apresentar um padrão de entrada, a rede avalia qual de seus nós corresponde mais de perto a essa entrada e, em seguida, os ajusta juntamente com seus nós vizinhos para melhorar ainda mais a correspondência (LEIJNEN; VEEN, 2020).

O método de classificação de padrões, segundo Francelin e Vieira (1994), no qual a Rede de Kohonen é baseada, é denominado Análise de *Clusters*. O conhecimento deste método facilita a compreensão do funcionamento da Rede de Kohonen. O método de classificação de Análise de *Clusters* foi proposto para solucionar problemas do tipo:

> “Dada uma amostra de **n** objetos, cada um deles representado por **p** variáveis, necessita-se de um esquema para agrupar os objetos em diferentes classes de modo que os similares fiquem em uma mesma classe”.

Um exemplo de uso desse tipo de rede neural é quando se deseja encontrar as classes verdadeiras. Por exemplo, em psiquiatria, havia um grande desacordo na classificação de pacientes depressivos, e a Análise de *Clusters* foi usada para definir grupos objetivos. Outra situação em que a Análise de *Clusters* pode ser útil é na redução dos dados. Como um exemplo, supõem-se que um grande número de cidades deve ser usado em um teste de mercado para um novo produto. Entretanto, para a realização do teste, um grande número de recursos está envolvido e seria conveniente usar um número menor de cidades sem prejudicar os resultados do teste. Desta forma, se as cidades forem agrupadas em um número pequeno de grupos de cidades similares, então um membro de cada grupo poderá ser usado para o teste de mercado (FRANCELIN; VIEIRA, 1994).

[1] Disponível em <https://kaiminghe.github.io/icml16tutorial/index.html>. Acesso em 28 maio 2024.

G.7 Redes de Cápsulas

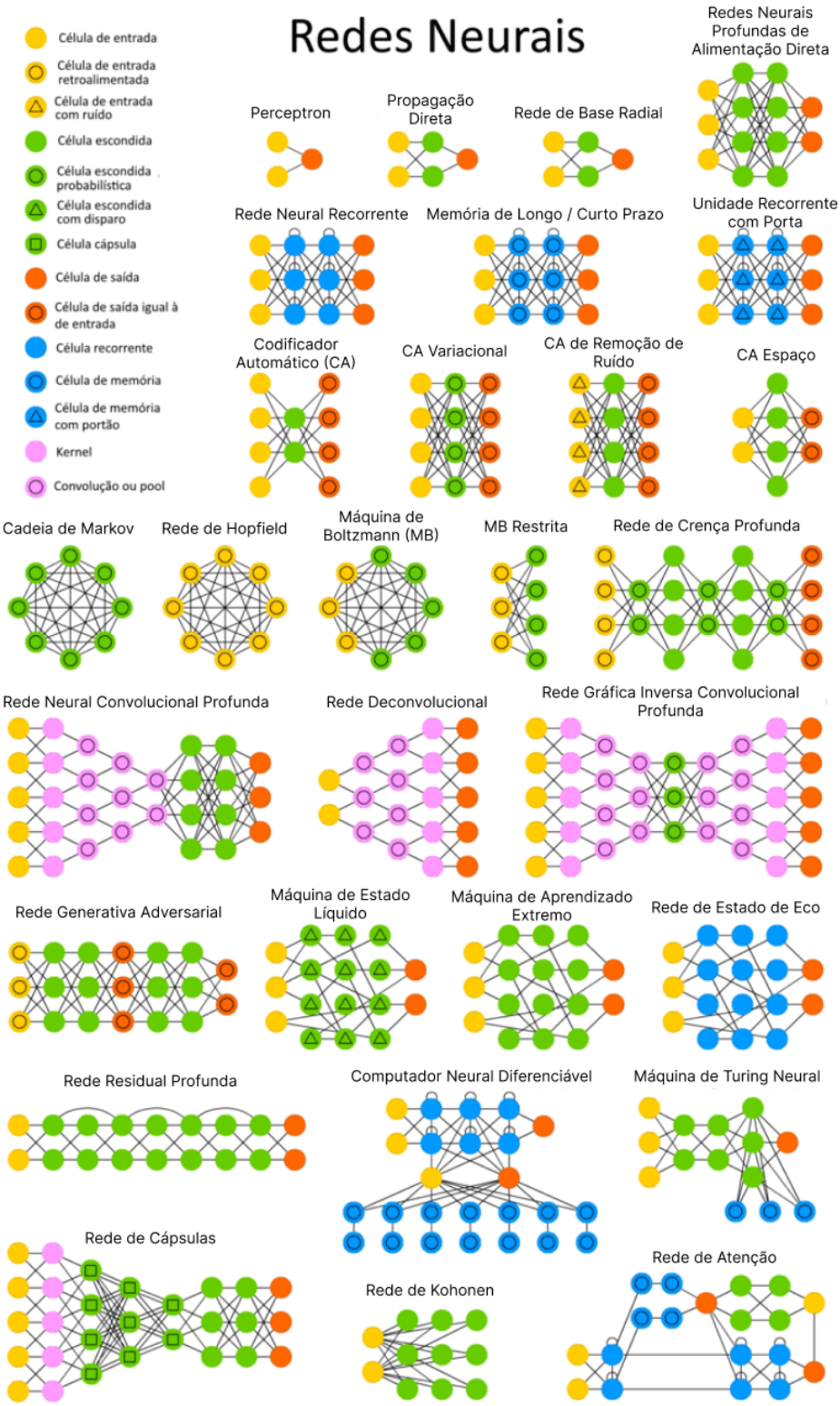
As Redes de Cápsulas, segundo Sabour, Frosst e Hinton (2017), oferecem uma alternativa biologicamente plausível às camadas de *pooling*. Os neurônios estão conectados com um vetor de peso em vez de um valor escalar. Isso permite que os neurônios transfiram simultaneamente vários tipos de informações, por exemplo, não apenas qual característica é detectada, mas também onde ela é detectada em uma imagem e qual é sua cor e orientação (LEIJNEN; VEEN, 2020).

Os algoritmos de aprendizado também são inspirados biologicamente pelo aprendizado de *Hebbian*, que valoriza previsões precisas da saída na próxima camada, segundo Leijnen e Veen (2020). Resumi-se a Teoria Hebbiana como "células que atuam juntas, permanecem conectadas". Essa teoria é muito utilizada para explicar aprendizagem associativa, na qual a ativação simultânea de células leva a um crescimento pronunciado na força sináptica. Tal aprendizado é conhecido como Aprendizagem Hebbiana.

Para melhor compreensão, de acordo com Oliveira (2020), pode-se entender a sinapse como sendo a região responsável pela comunicação entre dois ou mais neurônios, ou ainda entre um neurônio e um órgão que efetua a ação (ex. músculo ou glândula). Força sináptica é a força com a qual é executado o processo de transmissão de informação de um neurônio a outro ou de um neurônio a um órgão que efetua a ação. Nesse contexto, os neurônios estarem conectados por um vetor de peso potencializa esse efeito, permitindo transferir várias informações simultaneamente.

Visando conferir uma visão geral sobre os vários tipos de redes neurais existentes, a Figura 21 apresenta a estrutura de células que compõem cada uma das redes neurais citadas.

Figura 21 – Visão geral da arquitetura das redes neurais apresentadas



Fonte: Leijnen e Veen (2020) - Tradução (Autor)

Anexos

ANEXO A – FAQ com Informações do Termo de Uso

Figura 22 – Quem pode acessar

Fonte: Câmara dos Deputados (2024)

Figura 23 – Acesso aos dados

Fonte: Câmara dos Deputados (2024)