Universidade FUMEC
Faculdade de Ciências Empresariais
Programa de Pós-Graduação em Sistemas de Informação e Gestão do Conhecimento

# Detecção de solicitações de usuários duplicadas utilizando aprendizado de máquina

Márcio Afonso Cruz

Belo Horizonte
2021

Márcio Afonso Cruz

# Detecção de solicitações de usuários duplicadas utilizando aprendizado de máquina

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas de Informação e Gestão do Conhecimento da Universidade FUMEC como parte dos requisitos para a obtenção do título de Mestre em Sistemas de Informação e Gestão do Conhecimento.

**Área de concentração:** Gestão de Sistemas de Informação e do Conhecimento.

**Linha de pesquisa:** Tecnologia e Sistemas de Informação.

**Orientador:** Prof. Dr. Fernando Silva Parreiras

Belo Horizonte

2021

**Dados Internacionais de Catalogação na Publicação (CIP)**

C957d Cruz, Márcio Afonso, 1979-

Detecção de solicitações de usuários duplicadas utilizando aprendizado de máquina / Márcio Afonso Cruz. - Belo Horizonte, 2021.

78 f. : il.

Orientador: Fernando Silva Parreiras

Dissertação (Mestrado em Sistemas de Informação e Gestão do Conhecimento), Universidade FUMEC, Faculdade de Ciências Empresariais, Belo Horizonte, 2021.

1. Inteligência artificial. 2. Aprendizado do computador. 3. Processamento de linguagem natural (Computação). I. Título. II. Parreiras, Fernando Silva. III. Universidade FUMEC, Faculdade de Ciências Empresariais.

CDU: 681.3.72

Ficha catalográfica elaborada pela Biblioteca Universitária-FUMEC

UNIVERSIDADE FUMEC

Dissertação intitulada "**Detecção de solicitações de usuários duplicadas utilizando aprendizado de máquina**" de autoria de Márcio Afonso Cruz, aprovada pela banca examinadora constituída pelos seguintes professores**:**

Prof. Dr. Fernando Silva Parreiras – Universidade FUMEC
(Orientador)

Prof. Dr. Air Rabelo – Universidade FUMEC
(Examinador Interno)

Prof. Dr. Eric de Paula Ferreira – UEMG
(Examinador Externo)

Prof. Dr. Fernando Silva Parreiras
Coordenador do Programa de Pós-Graduação em Sistemas de Informação e Gestão do Conhecimento da Universidade FUMEC

Belo Horizonte, 21 de maio de 2021.

Air Rabelo

Fernando Silva Parreiras

Ricardo Terra

ABARIS

REQUESTED

| | |
|---|---|
| TITLE | **Assinatura de ata e contra-capas Universidade** |
| FILE NAME | **b5fabdc4-ceb3-49ad-b42b-7c1520670182.pdf** |
| RESQUEST ID | **signature_request_301bcaeb-22fb-40cb-9402-7ef8d** |
| RESQUESTED BY | **Karem Estefani Oliveira De Paula** |
| STATUS | **Completed** |

Professor (air@fumec.br)

| | | | |
|---|---|---|---|
| SENDED | 20/09/2021<br>21:33:01UTC±0 | SIGNED | 20/09/2021<br>21:34:09UTC±0<br>187.115.184.249 |

Professor (terra@ufla.br)

| | | | |
|---|---|---|---|
| SENDED | 21/09/2021<br>01:02:24UTC±0 | SIGNED | 21/09/2021<br>01:03:06UTC±0<br>177.66.53.2 |

Professor (fernando.parreiras@fumec.br)

| | | | |
|---|---|---|---|
| SENDED | 05/10/2021<br>20:56:26UTC±0 | SIGNED | 05/10/2021<br>20:56:46UTC±0<br>187.111.30.10 |

| | |
|---|---|
| COMPLETED | 05/10/2021<br>20:56:46 UTC±0<br>The document has been completed. |

# Resumo

Empresas de prestação de serviços de tecnologia adotam sistemas de gerenciamento de solicitações de usuários também chamados de *tickets*. Normalmente tais sistemas contam com muitos usuários e ocorre de diferentes usuários encontrarem e reportarem o mesmo *incidente*, *solicitação* e/ou *dúvida* gerando *tickets* duplicados nos repositórios. Nesse contexto, ocorre um trabalho dispendioso e redundante de classificação manual dos *tickets* tornando oneroso em termos de custo e tempo para intervenção manual.

Este trabalho avaliou o desempenho das técnicas de Aprendizado de Máquina utilizados na literatura para classificação e recuperação de *tickets* duplicados em língua estrangeira, e aplicou essas técnicas para detecção e recuperação de *tickets* duplicados em base de dados de língua portuguesa (Brasil).

Foi realizada a revisão da literatura com o objetivo de identificar as técnicas de Aprendizado de Máquina que apresentaram os melhores desempenho na tarefa de classificação e recuperação de *ticket* duplicados. As técnicas que atenderam esses critérios foram Naive Bayes, SVM, LSTM e BERT. A Base de dados que foi utilizada por esse trabalho pertence a uma empresa brasileira de prestação de serviços de tecnologia, terceirizadora de ativos e serviços para infraestrutura de TI (Locação de equipamentos e sistemas). A Empresa utiliza sistemas de gerenciamento de solicitações de usuários (*tickets*). Foram coletados 132.703 ticket contendo registros de Incidentes/Problemas, Perguntas/Dúvidas e Solicitações de Tarefas/Serviços, registrados no período compreendido entre Out/2019 e Set/2020.

Após a execução do experimento observou-se que o desempenho apresentado está próximo ao desempenho relatado na literatura para os modelos avaliados. Naive Bayes apresentou 53,30% de acurácia e 45,17% de precisão, SVM apresentou 63,87% de acurácia e 61,18% de precisão, LSTM apresentou 80,15% de acurácia e 73,64% de precisão e o melhor desempenho foram as abordagens baseadas em BERT (DistilBERT 78,47% de acurácia e 73,47% de precisão, BERT-Base 84,28% de acurácia e 81,00% de precisão e XML-RoBERTa 85,23% de acurácia e 82,58% de precisão sendo portanto o modelo que apresentou o melhor desempenho na tarefa de classificação e recuperação de *ticket* duplicados.

**Palavras chaves:** Inteligência Artificial, Aprendizado de Máquina, Processamento de Linguagem Natural, Recuperação de tickets, Classificação de tickets, Detecção de duplicatas, Solicitações duplicadas, tickets duplicados.

**Abstract**

Technology service providers adopt user request management systems also called *tickets*. Usually such systems have many users and different users find and report the same *incident*, *request* and/or *doubt* generating duplicate *tickets* in the repositories. In this context, there is an expensive and redundant work of manual classification of *tickets* making it costly and time-consuming for manual intervention.

This work evaluated the performance of *Machine Learning* techniques used in the literature for classification and retrieval of duplicate *tickets* in a foreign language, and applied these techniques to detect and retrieve duplicate *tickets* in a Portuguese database (Brazil).

A literature review was carried out in order to identify the *Machine Learning* techniques that presented the best performance in the task of classification and retrieval of duplicate *ticket*. The techniques that met these criteria were Naive Bayes, SVM, LSTM and BERT. The database used by this work belongs to a Brazilian company providing technology services, outsourcing assets and services for IT infrastructure (Lease of equipment and systems). The Company uses user request management systems (*tickets*). 132,703 tickets were collected containing records of Incidents/Problems, Questions/Doubts and Task/Service Requests, recorded in the period between Oct/2019 and Sept/2020.

After carrying out the experiment, it was observed that the performance presented is close to the performance reported in the literature for the evaluated models. Naive Bayes had 53.30% accuracy and 45.17% accuracy, SVM had 63.87% accuracy and 61.18% accuracy, LSTM had 80.15% accuracy and 73.64 % accuracy and the best performance were the BERT-based approaches (DistilBERT 78.47% accuracy and 73.47% accuracy, BERT-Base 84.28% accuracy and 81.00% accuracy and XML-RoBERTa 85.23% accuracy and 82.58% precision, being therefore the model that presented the best performance in the task of classification and retrieval of duplicate *ticket*.

**Keywords:** *Artificial Intelligence*, *Machine Learning*, *Natural Language Processing*, Ticket Recovery, Ticket Classification, Duplicate Detection, Duplicate Requests, Duplicate Tickets.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

***tickets*** Solicitações de Usuários. 5, 6, 13, 14, 16, 17, 22, 33, 34, 40, 43–49, 51, 52, 54–60, 63, 64, 67, 69–71

***ticket*** Solicitação de Usuário. 5–8, 13, 18–21, 28, 29, 33, 35–37, 40, 44, 45, 47, 50–52, 54–56, 59, 60, 67, 69

**AI** *Artificial Intelligence*. 6

**AM** Aprendizado de Máquina. 5, 13–16, 22, 23, 25, 31, 33, 36, 40, 46, 48, 49, 52, 59, 62, 69, 71

**AP** Aprendizado Profundo. 15, 23, 25, 30

**AS** Aprendizado Supervisionado. 25

**BERT** *Bidirectional Encoder Representations from Transformers (BERT)*. 31, 32

**CNN** *Convolutional Neural Networks - CNN*. 29

**CT** Classificação de Texto. 15, 33, 34

**DistilBERT** DistilBERT (destilado BERT). 32, 65, 68

**DL** *Deep Learning*. 11, 25, 71

**IA** Inteligência Artificial. 5, 13, 15, 25

**ITSM** *Information Technology Service Management (ITSM)*. 17

**LSTM** *Long Short-Term Memory - LSTM*. 30

**ML** *Machine Learning*. 6, 11, 22, 25, 35, 45, 52, 71

**NLP** *Natural Language Processing*. 6, 16, 33

**PLN** Processamento de Linguagem Natural. 5, 16, 23, 26, 29, 31–33, 65, 67, 71

**RNN** *Recurrent Neural Network - RNN*. 28

**RNR** Rede Neural Recorrente. 30

**roBERTa** *Robustly Optimized BERT Approach* (roBERTa). 32, 65, 68

**SLA** *Service Level Agreement*. 19, 21, 49

**SME** *Subject Matter Expert*. 19, 20, 57

**SPOC** *Single Point of Contact*. 18

**TCI** Tipo, Categoria, Item. 8, 20, 21

**TCSI** Tipo, Categoria, Serviço, Item ou Subcategoria. 8, 20, 21

**TFIDF** *TF and TF-IDF*. 55

# Sumário

# 1 Introdução

Buscar e manter a satisfação do cliente é uma das prioridades de empresas de prestação de serviços, o sistema de suporte técnico atua como ponto de ligação entre os usuários clientes e a equipe de suporte técnico da empresa [Roy et al., 2016].

Os agentes técnicos necessitam ler manualmente os *tickets*, classificá-los, responder com agilidade e precisão ou encaminhar ao próximo nível de suporte. Com uma base de clientes crescente, o tempo médio de espera por uma resposta pode prolongar-se, demandando tempo e trazendo custos elevados ao processo [Sasso et al., 2016, Eckstein et al., 2016].

As bases de dados geradas por essas interações entre clientes e o suporte técnico da empresas, constituem uma fonte propícia para a aplicação de Inteligência Artificial e Aprendizado de Máquina [Yang et al., 2016].

Um *ticket* anteriormente atendido poderá ressurgir pela mesma ou diferente razão. Nesse cenário, o mapeamento eficaz do *ticket* para um outro *ticket* solucionado anteriormente aumentará a eficiência do serviço de suporte técnico [Budhiraja et al., 2018].

A classificação e recuperação correta de um *ticket* para um outro *ticket* reportado anteriormente visa sobretudo o aumento da produtividade e a redução de custos. Tal ação visa evitar retrabalhos desnecessários por equipes técnicas na atuação de *tickets* com resoluções conhecidas e validadas [Kadar et al., 2011].

Na literatura há abordagens relatadas usando técnicas de Aprendizado de Máquina em repositório de *tickets* de solicitações e erros de língua inglesa [Son et al., 2014, Lee et al., 2015, Yang et al., 2016, Deshmukh et al., 2017], porém não foram encontrados trabalhos utilizando esses algoritmos em bases de dados de *tickets* na língua portuguesa (Brasil). Baseado nesse cenário têm-se o seguinte problema de pesquisa:

***Quais as técnicas de Aprendizado de Máquina utilizados na literatura para classificação e recuperação de tickets em bases de dados de língua estrangeira apresentam melhores desempenho na classificação e recuperação de tickets quando utilizadas em base de dados de língua portuguesa (Brasil)?***

A Proposta deste trabalho é identificar os algoritmos de Aprendizado de Máquina utilizados na literatura para classificação e recuperação de *tickets* em língua estrangeira, e aplicar tais algoritmos para detecção e recuperação de *tickets* duplicados em base de dados de língua portuguesa (Brasil).

## 1.1 Objetivos

Avaliar os resultados das técnicas de Aprendizado de Máquina utilizados na literatura para classificação e recuperação de *tickets* em bases de dados de língua estrangeira quando utilizados em base de dados de língua portuguesa (Brasil).

- **Objetivo Específico 1**: Identificar as técnicas de Aprendizado de Máquina utilizados na literatura para classificação e recuperação de *tickets* em língua estrangeira.
- **Objetivo Específico 2**: Aplicar as técnicas de Aprendizado de Máquina em uma base de dados de *tickets* de uma empresa brasileira de prestação de serviços em TI, na língua portuguesa.
- **Objetivo Específico 3**: Classificar as técnicas pela taxa de acurácia na detecção e recuperação de *tickets* duplicados.

## 1.2 Estrutura do documento

Este trabalho segue estruturado em 7 (sete) capítulos:

- **Capítulo 1**. Introdução - Apresenta a problematização que dá origem à pesquisa, seus objetivos, a justificativa de sua relevância, contempla a estruturação desta dissertação.

- **Capítulo 2**. Fundamentação Teórica - Apresenta a fundamentação teórica discorrendo sobre os temas que contribuíram para a criação dos construtos envolvidos no problema de pesquisa: Inteligência Artificial, Aprendizado de Máquina, Aprendizado Profundo, Redes Neurais e Classificação de Texto.

- **Capítulo 3**. Trabalhos Relacionados - São demonstrados os trabalhos encontrados na literatura relacionados com o estudo proposto.

- **Capítulo 4**. Metodologia - São abordados os procedimentos metodológicos adotados nesta pesquisa.

- **Capítulo 5**. Experimentos - Apresenta o experimento desenvolvido com a aplicação dos algoritmos selecionados, e por fim são apresentados e avaliados os resultados da execução de cada um dos modelos avaliados.

- **Capítulo 6**. Conclusões - São apresentadas as conclusões finais, limitações do experimento e sugestões de trabalhos futuros.

# 2 Fundamentação teórica

Conforme abordado nos capítulos anteriores, o problema a ser tratado neste trabalho aborda a automatização da detecção e recuperação de *tickets* duplicados.

Nas seções seguintes apresentaremos as técnicas utilizadas na implementação do experimento. Iniciaremos apresentando os conceitos e tipos de Aprendizado de Máquina com seus tipos de aprendizado e treinamentos de modelos preditivos [dos Santos and Gatti, 2014, Kim, 2014].

Na sequência apresentaremos o Processamento de Linguagem Natural (PLN), do inglês *Natural Language Processing* (NLP) que é uma área da ciência da computação, inteligência artificial e da linguística que estuda os problemas da geração e compreensão automática de línguas humanas naturais [Reshma and Remya, 2017]. Neste trabalho essas técnicas serão utilizadas para converter os textos das descrições dos *tickets* em uma representação estruturada que será utilizada pelos modelos preditivos de Aprendizado de Máquina [Greff et al., 2017].

## 2.1 Gestão de serviços de TI

A Gestão de serviços de TI do inglês *Information Technology Service Management (ITSM)*, tem por objetivo prover um serviço de TI com qualidade e alinhado às estratégias do negócio e necessidades dos clientes, objetivando a redução dos custos e eficiência operacional [Bon, 2002].

A área de TI das organizações que antes se concentrava exclusivamente no processamento de dados, atualmente tem em seu foco principal o planejamento e serviços em TI. A gestão de TI torna-se cada vez mais complexa e difícil pelo crescente número de ativos (*hardware*, *software* e recursos humanos), portanto elevar os fatores de eficiência e eficácia do sistema são necessários para suportar o ambiente em constante crescimento [Suhairi and Gaol, 2013].

Ao longo dos últimos anos, grandes empresas vêm implementando ITSM em seus departamentos de TI, porém, pequenas e médias empresas também têm a necessidade dessa implementação que deverá ser ajustada ao tamanho da empresa e ao seu modelo de gestão, para definição de até que ponto necessitará da assistência do ITSM [Gunawan, 2019].

Neste capítulo, será apresentado o fluxo de trabalho do primeiro nível (N1) de uma Central de Serviços, que efetua a classificação e realiza o atendimento inicial de solicitações e falhas reportados pelos usuários através dos *tickets*.

### 2.1.1 Central de serviços

A principal função da central de serviços é ser o único ponto de contato, em inglês *Single Point of Contact* (SPOC), entre os usuários e a equipe de suporte, centralizando nesse departamento todos os assuntos relacionados aos serviços de tecnologia da organização.

A Principal atividade da central de serviços é a gestão de incidentes e têm como objetivo garantir o reestabelecimento dos serviços de TI quando da ocorrência de uma interrupção não programada [Yandri et al., 2019].

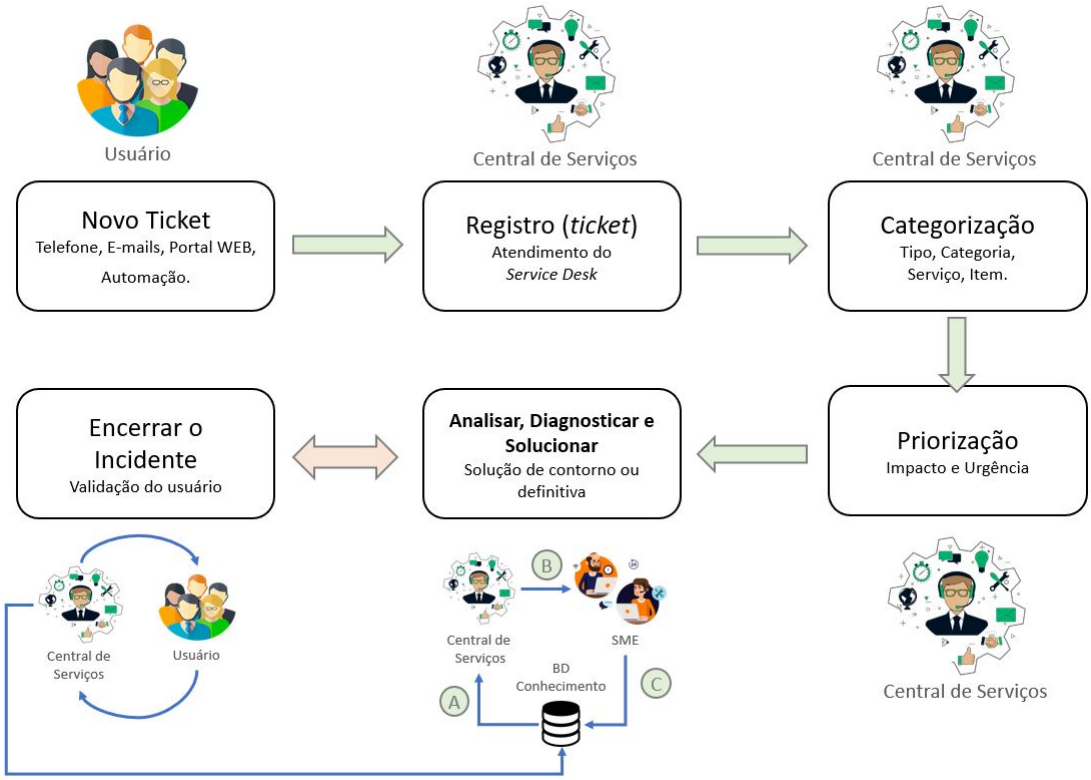
#### 2.1.1.1 Atividades da central de serviços

No atendimento de uma demanda de usuário, o atendente da central de serviços segue algumas atividades pré-definidas com o objetivo de reestabelecer integralmente a operação dos serviços afetados [Gunawan, 2019].

A Figura 1 ilustra o ciclo de vida de uma demanda recebida pela central de serviços e os papéis desempenhados neste processo.

Figura 1 – Ciclo de vida de um *ticket* de suporte

Adaptado de [Kallis et al., 2019]




- **Registro do *ticket*:**. A central de serviços efetua o registro do evento através de um *ticket* de *incidente*, *dúvida* ou *solicitação de serviço*, coletando todas as informações possíveis sobre a demanda a fim de auxiliar na classificação.

- **Categorização:** A central de serviços efetua a categorização do evento através da taxonomia TSCI – Tipo, Serviço, Categoria e Item.

- **Priorização:** A central de serviços prioriza o evento de acordo com o impacto definido no acordo de nível de serviço ou do inglês *Service Level Agreement* (SLA) e a urgência de acordo com o impacto causado pela falha.

- **Analisar, diagnosticar e solucionar:** A demanda é tratada com o objetivo de reestabelecer o serviço. A central de serviços segue os procedimentos descritos nos itens a seguir:

  – **(A)**. O atendente da central de serviços acessa a base de dados de conhecimento da organização e realiza uma busca de eventos anteriormente registrados que sejam semelhantes ou cópias exatas (duplicados) da demanda que está sendo tratada. Caso a busca retorne com sucesso, o atendente passa a demanda para a próxima atividade.

  – **(B)**. Caso o atendente não encontre uma solução para a demanda e não exista um procedimento para atender a solicitação, o mesmo irá direcionar o *ticket* para a fila do *Especialista no Assunto* do inglês *Subject Matter Expert* (SME).

  – **(C)**. O *Subject Matter Expert* (SME) irá efetuar o diagnóstico avançado e prover uma solução definitiva ou de contorno para a demanda do usuário. Se necessário, poderão ser envolvidos outros fornecedores, parceiros e/ou prestadores de serviços para auxiliar no atendimento da demanda. A solução encontrada será registrada na *Base de Conhecimento* da organização para futuras referências.

- **Encerrar o incidente:** A central de serviços irá validar a solução junto ao usuário. Caso a solução atenda às necessidades da demanda, o *ticket* será encerrado. Caso contrário, ele voltará para a fila do *Subject Matter Expert* (SME) com o registro das considerações do usuário.
  Após a validação final do usuário, a central de serviços irá realizar todos os procedimentos necessários para o encerramento do *ticket*.

#### 2.1.1.2 Classificação de demandas

A tarefa de classificação de um *ticket* é a ação de identificar o correto tipo de demanda e todos os componentes relacionados.

É necessário definir a prioridade e urgência da demanda, visando determinar a criticidade de acordo com o impacto que a indisponibilidade do serviço relacionado representa para o usuário.

##### 2.1.1.2.1 Categorização

A Atividade de categorizar o *ticket* normalmente ocorre no primeiro nível (N1) da *Central de Serviços*. O atendente realiza o entendimento da demanda analisando as informações passadas pelo usuário quando do registro do *ticket* [Marquis, 2010].

A Correta categorização de um *ticket* é necessária para que se possa solucionar a demanda diretamente no primeiro nível (N1) ou enviá-la para o SME correto [Rodrigues, 2012].

As atuais ferramentas de gestão de *ticket* passaram a utilizar o modelo de taxonomia Tipo, Categoria, Serviço, Item ou Subcategoria (TCSI) substituindo o modelo anterior de taxonomia Tipo, Categoria, Item (TCI) [Son et al., 2014].

Nessa evolução do modelo de categorização foi adicionado o campo **serviço** para categorização de demandas, utilizando informações originadas no catálogo de serviços. Dessa forma, o registo de um *ticket* passa a ser baseado no negócio da companhia e não apenas nos requisitos técnicos da tecnologia buscando indicar com clareza os serviços afetados [Montgomery and Damian, 2017].

A Tabela 1 mostra a descrição de um *ticket* real do *dataset* utilizado neste trabalho, registrado por um usuário através do portal WEB.

Tabela 1 – Descrição do *ticket*

| Descrição |
|---|
| "O Sinal da internet está muito lento,<br>apesentando falha na conexão,<br>o led vermelho está acesso no roteador." |

Fonte: Elaborada pelo Autor

A Tabela 2 mostra a categorização do *ticket* com a utilização do modelo Tipo, Categoria, Item (TCI) baseado em tecnologia.

Tabela 2 – Categorização de um *ticket* utilizando TCI

| Tipo | Categoria | Item (Subcategoria) |
|---|---|---|
| Falha | Rede | Hardware |

Fonte: Elaborada pelo Autor

A Tabela 3 mostra a categorização do *ticket* com a utilização do modelo Tipo, Categoria, Serviço, Item ou Subcategoria (TCSI) baseado em serviço.

Tabela 3 – Categorização de um *ticket* utilizando TCSI

| Tipo | Categoria | Serviço | Item (Subcategoria) |
|---|---|---|---|
| Falha | Rede | Rede Wireless Corporativa | Hardware |

Fonte: Elaborada pelo Autor

#### 2.1.1.2.2 Priorização

A atividade de priorização busca a correta identificação da importância relativa de um incidente ou solicitação de serviços. A prioridade é definida seguindo padrões de métricas de **urgência** e **impacto**, definindo dessa forma o SLA que é o tempo máximo previsto para que as ações sejam tomadas e o *ticket* seja solucionado. A métrica de **urgência** irá definir a velocidade com a qual é necessário resolver o *ticket*, enquanto a métrica de **impacto** irá avaliar o provável efeito que o incidente causará sobre o negócio da organização.

Figura 2 – Definição de prioridade de um *ticket*: impacto x urgência

| | | URGÊNCIA | | |
|---|---|---|---|---|
| | | 0 - Significativo | 1 - Moderado | 3 - Baixo |
| IMPACTO | 0 - Crítica | 1 | 2 | 3 |
| | 1 - Alta | 2 | 3 | 4 |
| | 3 - Média | 3 | 4 | 5 |
| | 4 - Baixa | 4 | 5 | 6 |

Fonte: Elaborada pelo Autor

## 2.2 *Machine Learning*

Aprendizado de Máquina (AM), do inglês *Machine Learning* (ML) é um subcampo da engenharia e da ciência da computação que deriva dos estudos de reconhecimento de padrões e da teoria do aprendizado computacional em inteligência artificial. Segundo a literatura, Aprendizado de Máquina é o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados [Samuel, 1959].

As técnicas de Aprendizado de Máquina têm sido utilizadas na análise de dados em vários campos do conhecimento tais como: administração, medicina, aplicações financeiras, educacionais, energéticas e outras. Essas técnicas possibilitam deduzir informações adicionais significativas daqueles dados processados pela mineração de dados. [Bulbul and Unsal, 2011]

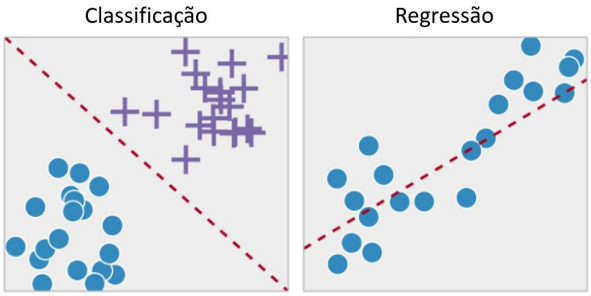
Trata-se de uma abordagem para aprender e analisar grandes volumes de dados, os algoritmos podem aprender com os dados sem depender da programação convencional, isto é, programação baseada em regras. Os algoritmos baseados em Aprendizado de Máquina permitem que grandes e complexos volumes de dados possam ser processados e analisados, sendo considerados a tendência na era da tecnologia da informação [Pushpa et al., 2017] .

Na sociedade atual, pode-se perceber que muitos aspectos são alterados de alguma maneira pelo Aprendizado de Máquina. Serviços de Streaming de filmes ou músicas sabem o que as pessoas gostam, e os buscadores conhecem o comportamento dos indivíduos pelo seu histórico. Essas ferramentas são valiosas aplicações considerando as grandes quantidades de informação a serem avaliadas em cada tomada de decisão [Beam and Kohane, 2018].

Deve-se distinguir os 4(quatro) tipos de tarefas de Aprendizado de Máquina:

- **Aprendizado supervisionado:** Essa é a categoria apresenta a questão como um problema de classificação binária, dado 2 (dois) *tickets* o classificador deve prever se são semelhantes ou não. A tarefa é encontrar uma função a partir de dados de treinamento rotulados (*tickets* classificados anteriormente) e prever rótulos desconhecidos em outros *tickets* (o conjunto de teste não rotulado). Se o rótulo é um número real, a tarefa chama-se *regressão*, se o rótulo vem de um conjunto finito e não ordenado a tarefa chama-se *classificação* [Greff et al., 2017].

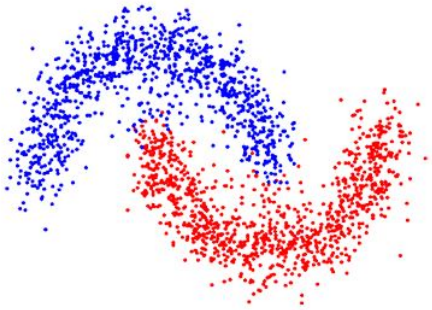
Figura 3 – Ilustração de regressão e classificação




Traduzido de [Greff et al., 2017]

- **Aprendizado não supervisionado:** Nessa categoria busca-se identificar grupos ou padrões a partir dos dados. Segundo a literatura houve o surgimento de algoritmos apropriados para treinar esses modelos que foram propostos para inúmeras tarefas em Visão Computacional, Processamento de Linguagem Natural, entre outros. Na área de Processamento de Linguagem Natural, os modelos de Aprendizado Profundo citados provaram ser superiores aos de abordagens clássicas de Aprendizado de Máquina em tarefas como identificação de voz, resposta a perguntas, análise de sentimentos e classificação de documentos.

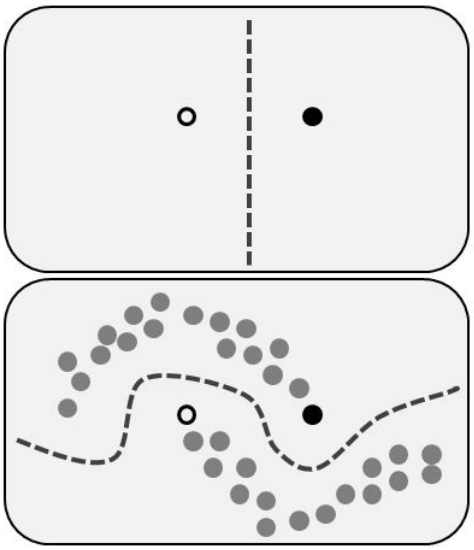
Figura 4 – Ilustração de aprendizado não supervisionado



[Greff et al., 2017, p. 2227]

- **Aprendizado semi-supervisionado:** Essa categoria inclui simultaneamente *Aprendizado supervisionado*e *Aprendizado não supervisionado*. A Tarefa utiliza dados rotulados e não-rotulados. Esse método é utilizado quando há dados rotulados e não rotulados, e há um esforço para rotular todos os dados. Essa abordagem permite melhorar a acurácia, pois utiliza dados não rotulados com dados rotulados [Lanyo and Wausi, 2018].

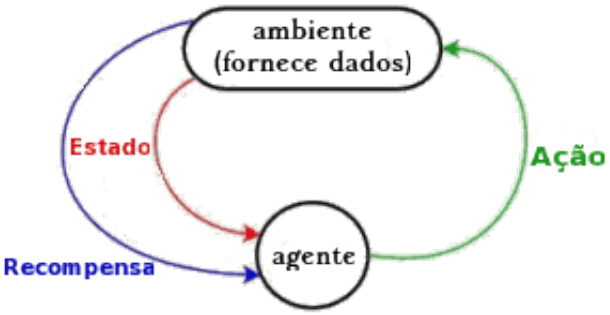
Figura 5 – Ilustração de aprendizado semi supervisionado



[Lanyo and Wausi, 2018, p. 89]

- **Aprendizado por reforço:** O Aprendizado por Reforço é uma técnica baseada no aprendizado pelo sucesso e fracasso, e fundamentada nos Processos de Decisão de Markov. Essa técnica difere de todas as tarefas anteriores porque não há conjunto de treinamento, rotulado ou não. O agente usa sensores para identificar o estado atual do ambiente, para em seguida tomar decisões. Para cada ação executada, o agente recebe um reforço. Essas informações são armazenadas e utilizadas nas escolhas das ações a serem realizadas. [Sutton and Barto, 1998, C. et al., 2016].

Figura 6 – Ilustração de aprendizado por reforço




Traduzido de [Sutton and Barto, 1998]

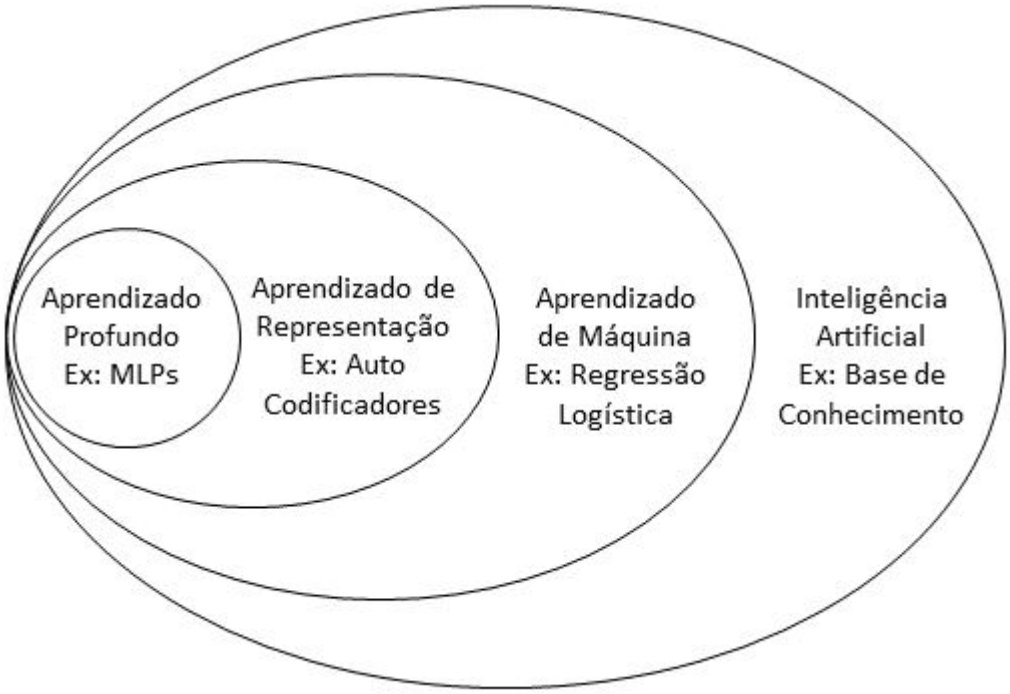
### 2.2.1 *Deep Learning*

O Aprendizado Profundo, do inglês *Deep Learning* é um segmento do *Machine Learning* baseado em um conjunto de algoritmos que buscam modelar abstrações de alto nível de dados usando um grafo profundo com várias camadas de processamento, compostas por sua vez de várias transformações lineares e não lineares. Provê uma estrutura poderosa para o Aprendizado Supervisionado [LeCun et al., 2015].

O Aprendizado de Máquina permite que modelos computacionais compostos por várias camadas de processamento aprendam representações de dados com vários níveis de abstração. Esses métodos contribuíram na execução de atividades com reconhecimento de fala, reconhecimento visual de objetos, detecção de objetos e outros domínios, como descoberta de medicamentos e genômica. A Técnica descobre uma estrutura complexa atuando em grandes conjuntos de dados usando o algoritmo de retro propagação para indicar como uma máquina deve alterar seus parâmetros internos que são usados para calcular a representação em cada camada da representação na camada anterior [Goodfellow et al., 2016].

O Diagrama de Venn mostra como o Aprendizado Profundo é um tipo de aprendizado de representação, que por sua vez, é um tipo de *Machine Learning*. Cada seção do diagrama de Venn inclui um exemplo de uma tecnologia de Inteligência Artificial.

Figura 7 – Ilustração da intersecção do aprendizado de máquina com outros campos da inteligência artificial




Adaptado de [Goodfellow et al., 2016]

### 2.2.2 Redes neurais artificiais

A ideia das Redes Neurais Artificiais foi introduzida na literatura pelo neurofisiologista Warren McCulloch e o matemático Walter Pitts ao escreveram um artigo sobre como os neurônios poderiam funcionar e para isso, modelaram uma rede neural simples usando circuitos elétricos [McCulloch and Pitts, 1943].

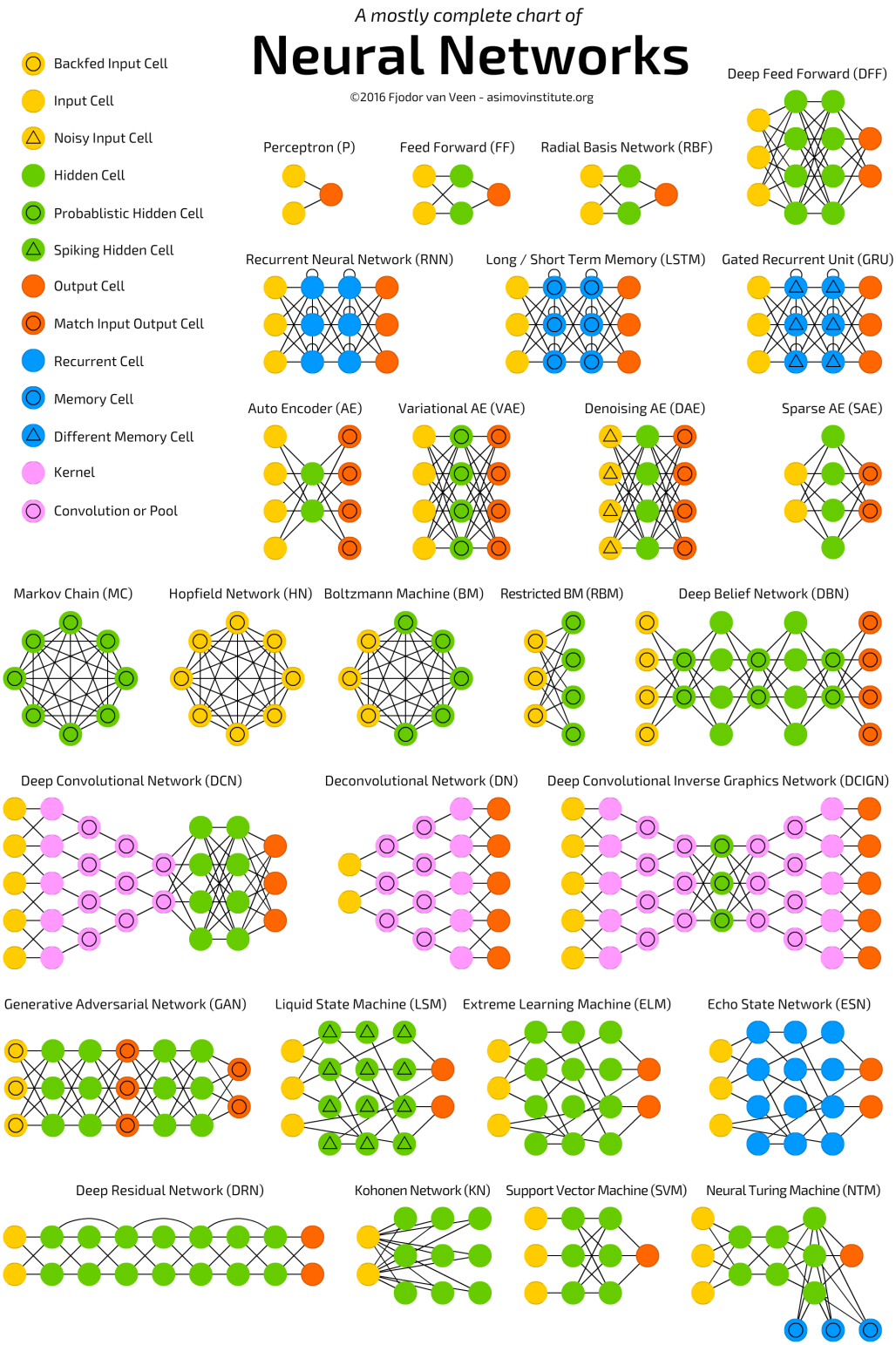
Os Pesquisadores Warren McCulloch e Walter Pitts desenvolveram um modelo para Redes Neurais baseadas em matemática e algoritmos denominado de Lógica de Limiar *threshold logic*. Com base nesse modelo, a abordagem de rede neural seguiu dois caminhos distintos: um voltado para processos biológicos no cérebro, e o outro com ênfase na aplicação de redes neurais à inteligência artificial.

Baseado nesses estudos, conclui-se que os neurônios possuem dois estados, o de ativo e inativo. O neurônio torna-se ativo quando sua saída supera um limiar estabelecido. A Partir desse conceito, surgiram na literatura outras abordagens, como as Redes Neurais *Feedforward*, também conhecidas como Perceptron de Múltiplas Camadas.

As redes Perceptron são utilizadas em aplicações comerciais, que também utilizam outros tipos especializados dessa abordagem. Alguns exemplos são as Redes Neurais Convolutivas e as Redes Recorrentes, usadas em aplicações de Processamento de Linguagem Natural [Bengio, 2009, Goodfellow et al., 2016].

o pesquisador Fjodor Van, em sua pesquisa sobre topologias de redes neurais artificiais, realizou um compilado do desenho dessas redes e o Instituto Asimov publicou uma cartilha demonstrando essas informações. O Instituto Asimov é um instituto de pesquisa de IA, sem fins lucrativos, que explora a relação entre aprendizado profundo e criatividade. Um dos objetivos do instituto é que as redes neurais artificiais gerem produtos, conteúdo, sugestões, estilos e ideias.

Figura 8 – Topologias de redes neurais artificiais



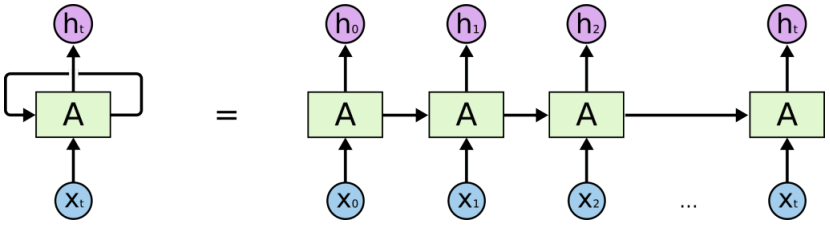

Fonte: Asimov Institute
https://www.asimovinstitute.org/

#### 2.2.2.1 *Redes Neurais Recorrentes - RNN*

As Redes Neurais Recorrentes, do inglês *Recurrent Neural Network - RNN* utilizam as informações em uma sequência do texto, tendo como base que o texto é uma sequência de palavras. Nessa sequência de palavras, uma representação numérica (vetores GloVe ou Word2Vec) da palavra é conectada a uma rede neural e sua saída é calculada. Enquanto é executado o cálculo de saída para a próxima palavra, a saída da palavra anterior também é considerada no cálculo. As RNN são denominadas recorrentes porque executam o mesmo cálculo para cada elemento de uma sequência usando a saída de cálculos anteriores. Os termos de viés são separados e poderão ser adicionados posteriormente. RNN (ti) é a saída no i-ésimo, que pode ser utilizado como está ou pode ser alimentado novamente para uma construção parametrizada [Bishop, 2006].

Uma das característica de uma RNN é que suas conexões possibilitam reter a memória das conexões de entradas anteriores, que são armazenados internamente na memória da rede e influenciam os resultados de saída. A figura 9 demonstra essa característica, que se faz importante quando os dados de entrada tem relações de dependências entre si [Goodfellow et al., 2016].

Figura 9 – Exemplo de uma rede neural recorrente

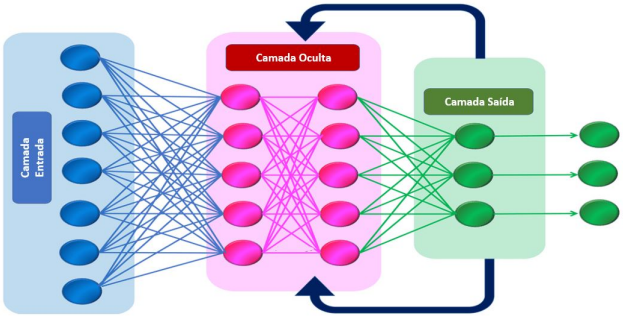



[Goodfellow et al., 2016, p. 7]

Os cálculos matemáticos dessas redes são aprimorados pela entrada repetitiva de dados semelhantes, de forma a se assemelhar à forma de conversação humana. Dessa forma as RNNs consideram a entrada como *informação conhecida*.

Como exemplo, um usuário insere um novo *ticket* no sistema, após a análise do *ticket* a RNN analisa as palavras, reconhecendo os traços de cada palavra e toda a estrutura da sentença por meio do texto do *ticket*, bem como as estruturas gramaticais presentes. Portanto, esses modelos possuem dois tipos de entradas: o do passado (dados codificados) e o do presente (dados de entrada), que determinam o nível de recorrência que as redes devem responder a novas entradas [Yalur, 2019].

Figura 10 – Camadas de uma rede neural recorrente



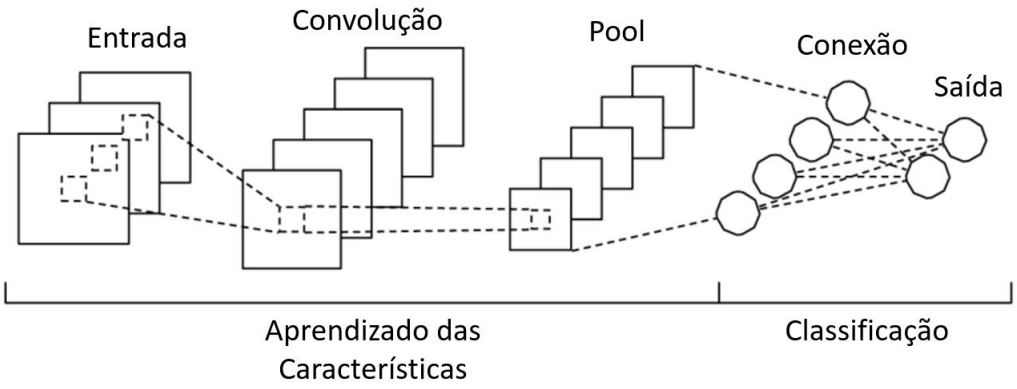

Traduzido de [Yalur, 2019]

#### 2.2.2.2 *Redes Neurais Convolucionais - CNN*

As Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks - CNN* utilizam camadas com filtros de convolução que são aplicados a recursos locais. Originalmente concebidos para a visão computacional, os modelos da CNN demonstraram ser eficazes para Processamento de Linguagem Natural e alcançaram resultados em análise semântica, recuperação de consulta de pesquisa, modelagem de sentenças e outras atividades tradicionais de Processamento de Linguagem Natural [Lecun et al., 1998, Yih et al., 2011, Collobert et al., 2011, Shen et al., 2014, Kalchbrenner et al., 2014].

Um filtro convolucional pega parte do texto de entrada, calcula uma função, retorna o valor calculado; na sequência, pega outra parte do texto, calcula a mesma função e retorna o valor calculado. O Filtro repete esse processo até que o texto seja calculado. Todas os resultados são calculados para formar um representação menor da entrada ou uma camada de pool é aplicada. Uma camada de pool agrega o máximo de saídas e a camada média do pool carrega a média das saídas. Ao se aplicar à um processamento de imagens, o cálculo ocorre em duas dimensões 2D porque as imagens estão em 2D. No contexto do texto do *ticket*, têm-se uma entrada unidimensional que é a concatenação da representação numérica das palavras do texto. O filtro convolucional começa no início do texto, considerando um tamanho do texto fixo (denominado Tamanho da janela ou Tamanho do filtro), executa um cálculo de função com o texto de entrada e gera o valor. O Operador move o índice para a direita em uma certa quantidade (chamada comprimento da passada) e repete o mesmo cálculo. O processo continua até o final do texto. As saídas são concatenadas ou uma é aplicada a camada de pool. O mesmo operador pode ser aplicado novamente (com o mesmo grupo de parâmetros ou ajustados), considerando a saída reduzida como entrada. Essas arquiteturas convolucionais empilhadas são chamadas Redes convolucionais profundas. A codificação final gerada por essas arquiteturas pode ser usado para a tarefa de previsão final do *ticket* [Deshmukh et al., 2017].

Figura 11 – Arquitetura de uma rede CNN




Adaptado de [Deshmukh et al., 2017]

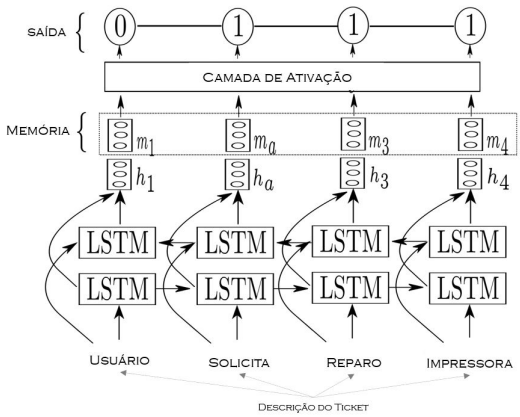
#### 2.2.2.3 *Long Short-Term Memory - LSTM*

A Arquitetura Memória de Longo Prazo, do inglês *Long Short-Term Memory - LSTM* é uma arquitetura artificial de Rede Neural Recorrente usada no campo do Aprendizado Profundo. Ao contrário das redes neurais de avanço padrão, o LSTM possui conexões de retro-alimentação. Essa arquitetura é capaz de processar, além de dados de pontos únicos (como imagens), também sequências inteiras de dados (como fala, vídeo ou textos longos) [Hochreiter and Schmidhuber, 1997].

Uma das arquiteturas base de uma LSTM é composta por uma célula (a parte da memória) e três *reguladores*, denominados *portões*, do fluxo de informações dentro da unidade LSTM: uma porta de entrada, uma porta de saída e uma porta para esquecer. Existem variações da unidade LSTM que não possuem um ou mais desses portões e existem variações que tenham outros portões. Por exemplo, as unidades recorrentes fechadas (GRUs) não possuem uma porta de saída [Shah and Patel, 2016].

A Principal característica da LSTM é o estado da célula, ela possui a capacidade de alterar e guardar o conteúdo na célula dependendo da tarefa a ser executada utilizando dos portões e estados. Uma execução de LSTM *forward* é aquela que recebe a sequência de entrada, A LSTM *backward* é aquela que recebe na entrada a ordem inversa. Uma LSTM *backward* é utilizada para capturar as dependências de uma palavra em futuras palavras na sequência original. Recebe o nome bi-LSTM a junção formada por uma rede LSTM *forward* com uma LSTM *backward*, esse modelo será utilizado no experimento desse projeto [Deshmukh et al., 2017].

Figura 12 – bi-LSTM codificando um ticket



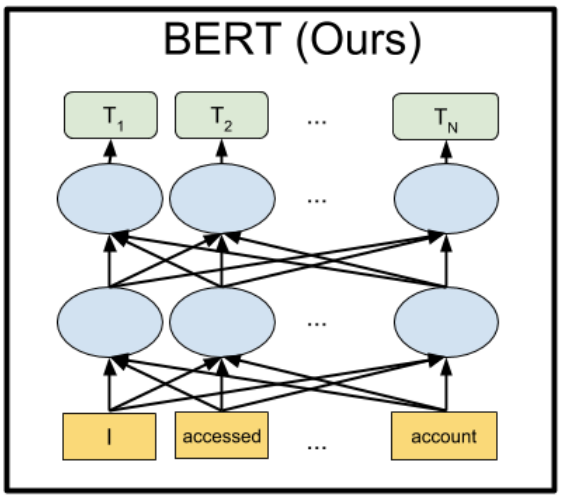

Adaptado de [Deshmukh et al., 2017]

#### 2.2.2.4 *BERT*

*Bidirectional Encoder Representations from Transformers (BERT)* é uma técnica de Aprendizado de Máquina baseada em transformador para pré-treinamento de Processamento de Linguagem Natural desenvolvida pelo Google. O BERT foi criado e publicado em 2018 por Jacob Devlin e seus colegas do Google. Desde 2019, o Google tem aproveitado o BERT para entender melhor as pesquisas dos usuários [Devlin et al., 2018].

A arquitetura utilizada pelo *BERT* foi construída para que ao se treinar e ao se predizer os *embeddings*, tanto as palavras anteriores quanto seguintes são consideradas, e o mecanismo armazena as informações das palavras mais e menos relevantes na frase. Uma representação da arquitetura de funcionamento do *BERT* está ilustrada na Figura 13.

Figura 13 – Representação da arquitetura do *BERT*




[Devlin et al., 2018, p. 123].

#### 2.2.2.5 *DistilBERT*

DistilBERT (destilado BERT) é uma versão pré-treinada menor e mais rápida de propósito geral do BERT, que retém quase as mesmas capacidades de compreensão da linguagem [Sanh et al., 2019]. São utilizados modelos de linguagem pré-treinados com destilação de conhecimento (distilation), uma técnica de compressão em que um modelo compacto é treinado para reproduzir o comportamento de um modelo maior ou um conjunto de modelos, resultando em modelos mais leves e rápidos no tempo de inferência, enquanto também requer menor treinamento computacional. A técnica de destilação [Hinton et al., 2015] consiste em treinar um modelo, baseado em um modelo maior, denominado professor, que é usado para ensinar o modelo destilado, denominado aluno, a reproduzir o comportamento do modelo maior. Assim, DistilBERT é um modelo mais enxuto baseado no comportamento do modelo BERT original.

#### 2.2.2.6 *RoBERTa*

*Robustly Optimized BERT Approach* (roBERTa) é um *framework* baseado em BERT para pré-treinamento de modelo de linguagem que estende BERT treinando o modelo com *batches* maiores, maior volume de dados, e em sequências mais longas, também removendo a tarefa de previsão da próxima frase e alterando dinamicamente o padrão de mascaramento aplicado aos dados de treinamento [Liu et al., 2019]. Resultados experimentais em tarefas PLN usando os *benchmarks* GLUE, RACE e SQuAD mostram que roBERTa atinge resultados no estado da arte, superando BERT e XLNet, uma abordagem de aprendizagem autorregressiva [Yang et al., 2019].

## 2.3 Processamento de linguagem natural

Processamento de Linguagem Natural (PLN), do inglês *Natural Language Processing* (NLP) é uma área da ciência da computação, inteligência artificial e da linguística que estuda os problemas da geração e compreensão automática de línguas humanas naturais. São objetivos dessa área a compreensão e processamento da linguagem natural. Por exemplo, a tradução automática, que se concentra na conversão de texto de uma linguagem humana para outra automaticamente [Reshma and Remya, 2017].

### 2.3.1 Categorização de texto

Uma das atividades em Aprendizado de Máquina é a classificação de documentos, em que cada instância representa um documento e a classe da instância é o tema do documento [Mubeen et al., 2011]. Os documentos são diferenciados de acordo com as palavras presentes neles.

A Classificação de Texto é uma atividade de mineração de texto que tem por finalidade permitir que usuários possam extrair informações a partir de recursos textuais, lidando com operações como, recuperação, classificação (supervisionada, não supervisionada) e sumarização utilizando técnicas de Processamento de Linguagem Natural, Mineração de Dados e Aprendizado de Máquina [Wang et al., 2015].

A Classificação de Texto, também conhecida como classificação de tópicos, é uma metodologia para classificar parte do texto em classes. Essa técnica vem sendo utilizadas por empresas na atividade do usuário redigir os campos necessários para o registro do *ticket* no sistema automatizado, dessa forma o sistema realiza o redirecionamento do *ticket* do usuário para o departamento correto. Nessa situação, um classificador de texto pode classificar esses problemas, o que pode economizar tempo e esforços. Na automação industrial, os defeitos registrados pelas unidades de negócios locais (LBU) e utilitários estão no formato de texto [Parmar et al., 2018].

Na atividade de Classificação de Texto, são executadas as seguintes fases [Venkatesh and Ranjitha, 2018, Lanyo and Wausi, 2018]:

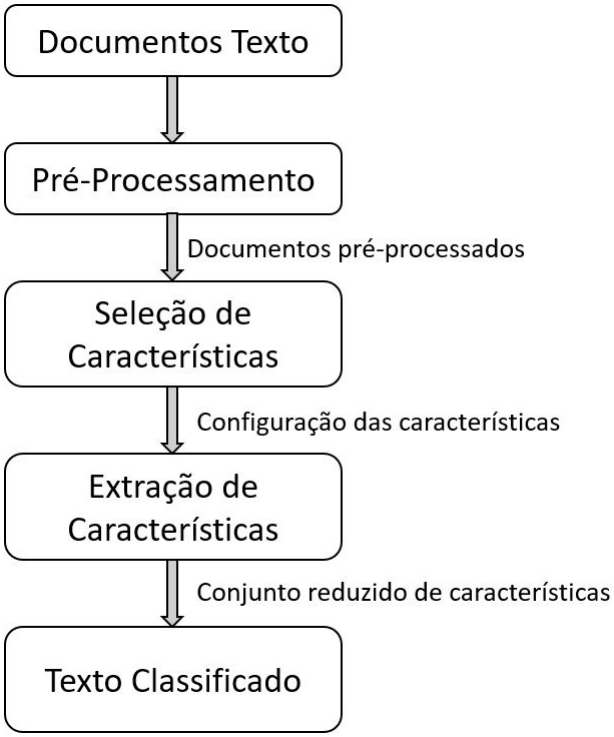
- **1. Coleta de documentos:** Fase em que são obtidos os dados a serem utilizados na fase de treinamento para construção dos classificadores.
- **2. Pré-processamento:** Fase em que são preparadas a estruturação dos dados para se obter as informações do *ticket*, a partir de um conjunto de informações desestruturados contidas no texto do *ticket*.
- **3. Fase de treinamento:** Fase em que o Aprendizado de Máquina é utilizado nos dados dos *tickets* que já foram previamente categorizados de forma manual, com o

objetivo de extrair informações que auxiliem na categorização automática de novos *tickets*.

- **4. Classificação:** Fase em que os novos *tickets* ainda não rotulados são organizados nas categorias preexistentes a partir da aprendizagem adquirida.

A Figura 14 demonstra o fluxo de atividades realizadas na Classificação de Texto.

Figura 14 – Fluxo de classificação de texto




Fonte: Elaborada pelo Autor

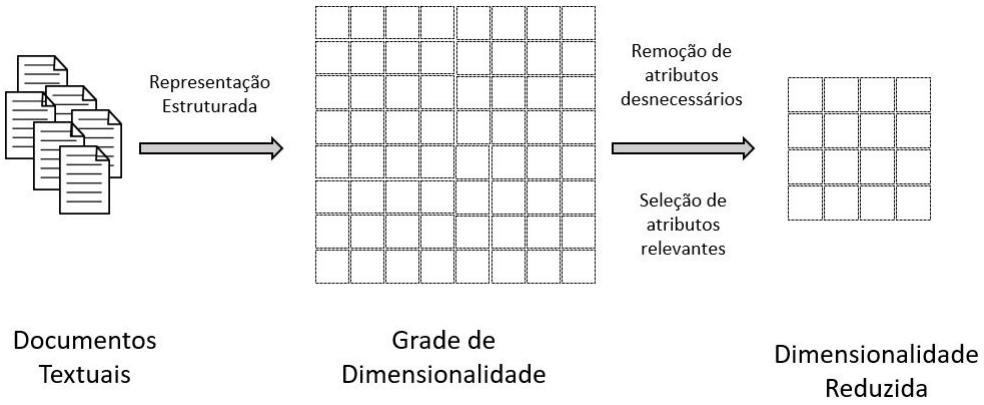
### 2.3.2 Pré-processamento

Após a realização da coleta de dados, a próxima atividade é a preparação dos textos, para que possa ser manipulado pelos algoritmos de mineração de texto. Essa etapa é denominada *Pré-Processamento* sendo responsável por converter o texto de entrada em uma representação estruturada, a ser utilizada pelos algoritmos de *Machine Learning* [Shah and Patel, 2016].

Nessa etapa de *Pré-processamento* é criado o modelo de representação dos documentos, transformando textos em linguagem natural em dados estruturados. Na literatura há modelos de representação estruturada de documentos textuais, Utiliza-se um algoritmo de categorização de texto como relatado o Modelo de Espaço Vetorial ou do inglês Vector *Space Model(VSM)* [Qamili et al., 2018].

Após a criação do modelo de representação de textos, o resultado deverá estar apto a ser encaminhado para os algoritmos de *Machine Learning*. Para que isso seja possível são realizadas atividades de análise de dados que irão selecionar apenas as características que melhor expressem o conteúdo do texto [Zhang et al., 2019].

Figura 15 – Representação estruturada do texto




Adaptado de [Zhang et al., 2019]

### 2.3.3 Análise léxica

Análise léxica é o processo de analisar a entrada de texto validando com o alfabeto e retirando os itens que não são significativos, reduzindo o texto e o tempo de processamento dos algoritmos que trabalharão com o texto [Guoxiang and Linlin, 2011].

Durante o processo de análise léxica os dígitos e sinais de pontuação são desconsiderados, os termos são isolados e converte-se o texto para maiúsculas.

Essa atividade, visa gerar agilidade no algoritmo de indexação.

Considere a seguinte frase, retirada do campo *descrição* de um *ticket* do *dataset* do experimento:

**"Verificar contas de e-mails da Diretoria de Documentação e Registro"**

A Análise léxica consiste no tratamento e limpeza dos textos, portanto a frase resultante da aplicação desta etapa encontra-se sem o hífen e o ponto final, devido a esses não serem relevantes para o texto. Nessa etapa, retiram-se também os acentos.

**VERIFICAR CONTAS DE EMAILS DA DIRETORIA DE DOCUMENTACAO E REGISTRO**

### 2.3.4 Tokenização

A tokenização é utilizada para decompor o documento em cada termo que o compõe. Os delimitadores utilizados para tokenização são: o espaço em branco entre os termos, quebras de linhas, tabulações, e alguns caracteres especiais [Aho et al., 2007].

Na execução da tokenização em linguagens delimitadas por espaço, as ambiguidades existem entre o uso de sinais de pontuação, tais como ponto final, vírgulas, aspas, apóstrofos e hifens, uma vez que o mesmo sinal de pontuação pode servir para diferentes funções em uma mesma sentença. Considere o texto exemplo de um *ticket*:

**"Usuário informou que o sistema de medição Nº. 93 - XPTO SERVICOS FINANCEIROS S/A não está buscando a linha de serviço 2 para a aba de (medição) Linha 2 no valor de R$ 5.000,00 Gentileza verificar."**

O texto é uma descrição de um *ticket* do *dataset* deste projeto, nele encontramos o ponto final de três formas distintas: junto aos números para separação de classes numéricas (milhares em 5.000,00), em abreviações (Nº.) e para marcar o final da sentença. Nesta última maneira, se percebe uma ambiguidade: o ponto final vem após a palavra **verificar**, o que poderia confundir com o ponto separador de classes numéricas. O tokenizador deve, assim, estar atento ao uso dos sinais de pontuação e determinar quando um sinal de pontuação é parte de outro *token* e quando é separado de um *token*.

Para resolver esses problemas, devem ser feitas decisões de tokenização sobre o texto. Por exemplo, deve-se decidir se trataria a frase “R$ 5.000,00” diferentemente do que se tivesse sido escrita como “5 mil reais” ou “R$ 5 mil”.

### 2.3.5 Remoção de *stopwords*

*Stopwords* (ou palavras de parada – tradução livre) são palavras que podem ser consideradas irrelevantes para a recuperação do conhecimento nos textos. Após a execução da atividade de tokenização, a próxima atividade a ser realizada é a identificação dos segmentos do texto que podem ser desconsiderados para as próximas atividades de Aprendizado de Máquina.

No texto do *ticket* há *tokens* que possuem reduzido valor semântico, portanto não

são úteis para o entendimento e compreensão da solicitação do usuário. Esses *tokens* de pouco valor são considerados *stopwords* e seu conjunto é chamado de *stoplist* em um sistema de mineração de textos [Gunasekara and Haddela, 2018].

As palavras irrelevantes são gerenciadas por meio de objetos denominados listas de palavras irrelevantes *stoplists*. Essa lista quando associada a um texto completo do *ticket* é utilizada para a remoção dessas palavras do texto do *ticket*.

***Nobreak* da sala do 21 andar apitando**

No exemplo, o texto final do *ticket*, após a retirada das *stopwords* ficará:

**Nobreak sala 21 andar apitando**

Exemplos de *stopwords* são: de, a, o, que, e, do, da, em, um, para, é, com, não, uma, os, no, se, na, por, mais, as, dos, como, mas, foi, ao, ele, das, tem, à, seu, sua, ou, ser, quando, muito, há, nos, já, está, eu, também, só, pelo, pela.

Figura 16 – Identificação de *stopwords*

... ~~Na~~ maioria ~~das~~ vezes ~~os~~ documentos retornados ~~pelas~~ ferramentas ~~de~~ recuperação ~~de~~ informações evolvem ~~um~~ contexto ~~mais~~ amplo fazendo ~~com~~ ~~que~~ ~~o~~ usuário tenha ~~que~~ garimpar ~~ou~~ ~~seja~~ especificar ~~ou~~ filtrar estes documentos ~~o~~ ~~que~~ demanda tempo ~~e~~ conhecimento ~~a~~ fim ~~de~~ obter ~~a~~ informação ~~que~~ ~~ele~~ realmente necessita ...

[Aranha et al., 2007, p. 172]

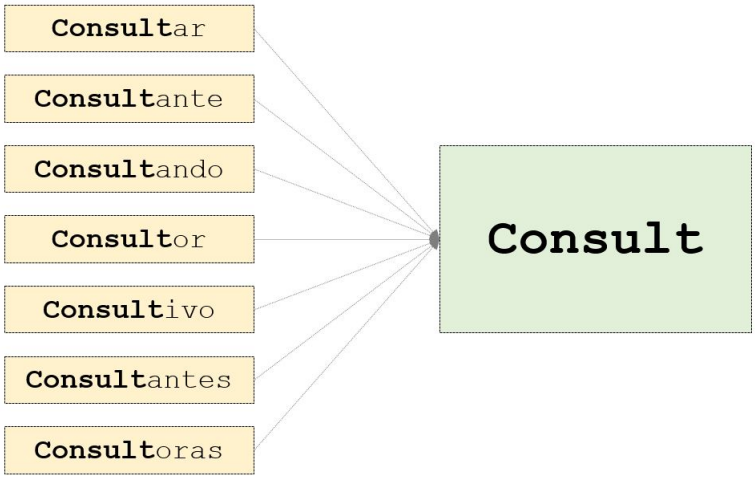
### 2.3.6 Stemização (em inglês *Stemming*) ou Radicalização

Radicalização é o processo de combinar as diferentes formas de uma palavra em uma representação única recuperando o radical (stem) [Orengo and Huyck, 2001].

O Radical é uma sequência de caracteres que resulta do processo de radicalização, pode não ser idêntico à raiz linguística, porém permite tratar da mesma forma as variações de uma mesma palavra. As palavras **Computador** e **Computadores** são essencialmente idênticas, porém sem o processo de radicalização serão tratadas como palavras distintas [Dias and Malheiros, 2005].

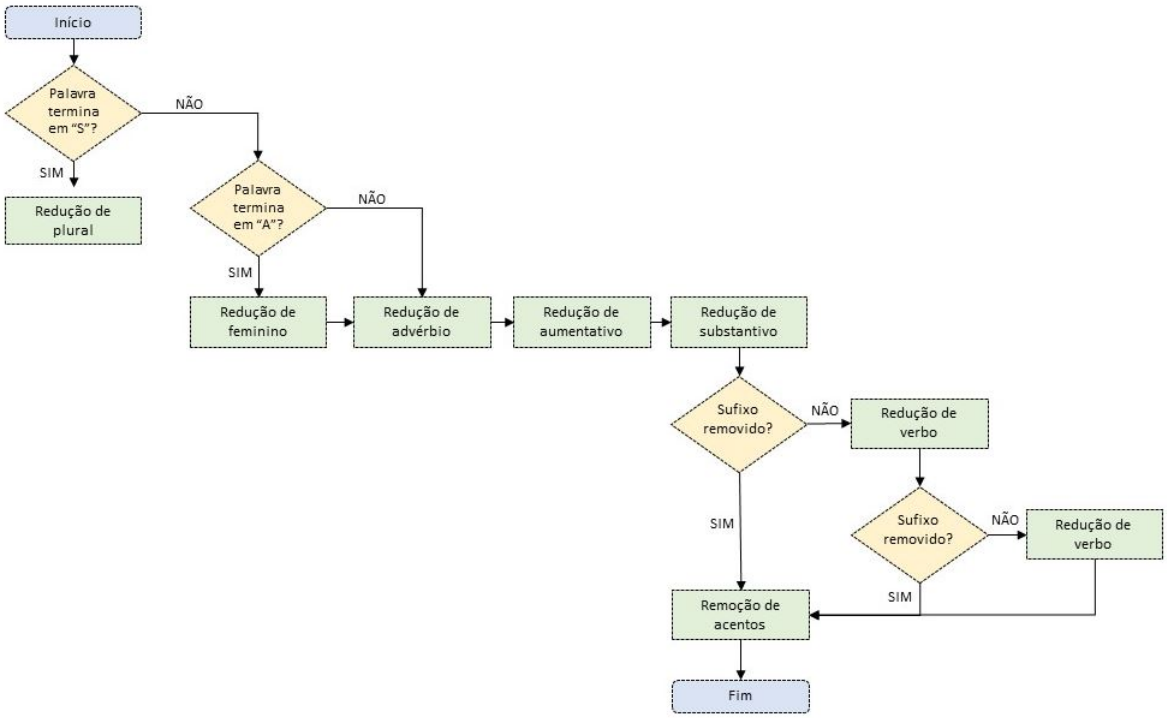
Um dos radicalizadores utilizados tradicionalmente na literatura é o Radicalizador de Porter que foi desenvolvido especificamente para a Língua Inglesa. Tal técnica mostrou baixa aderência à Língua Portuguesa, porém, a implementação do algoritmo *Portuguese Stemmer* proposto por Viviane Orengo e Christian Huyck demonstrou alta aderência para a Língua Portuguesa [Van Rijsbergen et al., 1980, Orengo and Huyck, 2001].

Figura 17 – Exemplo de radicalização




Traduzido de [Orengo and Huyck, 2001]

Figura 18 – Passos do algoritmo de radicalização




Traduzido de [Dias and Malheiros, 2005]

### 2.3.7 *Word Embeddings*

Constitui um desafio em aplicação de modelos de aprendizado de máquina para processamento de linguagem natural a localização de uma representação numérica correta para as palavras. Existem múltiplos modelos para representar uma palavra como um vetor numérico, com base no contexto em que aparece. Esse modelos vetoriais para palavras têm sido utilizado de várias formas, incluindo Análise Semântica latente, do Inglês *Latent Semantic Analysis (LSA)* [Haykin, 1994].

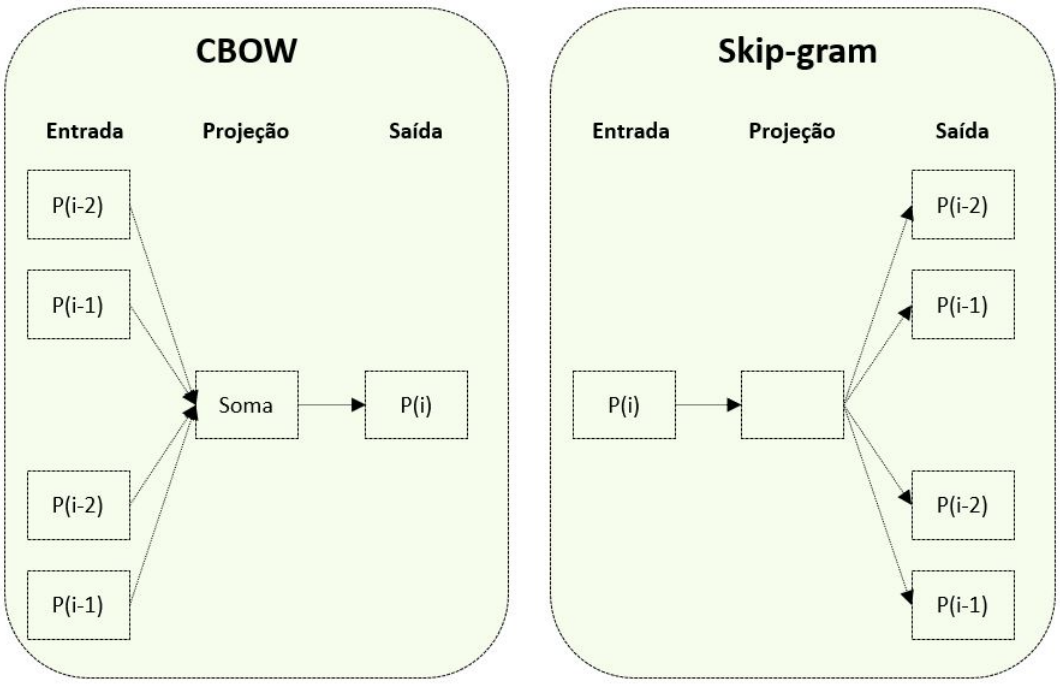
A Representação das palavras como vetor numérico no contexto de redes neurais foi proposta anteriormente na literatura. Neste trabalho, para cada palavra no vocabulário é atribuído um recurso de palavra distribuída de vetor. A distribuição de probabilidade de sequências de cada palavras é calculada e expresso em termos destes vetores de recurso de palavra [Bengio et al., 2003].

Os vetores e parâmetros de probabilidade da função (que é uma rede neural) são aprendidos por treinamento em uma rede neural *feed-forward* [Dumais, 2004].

Inspirado nesse modelo, outros trabalhos propuseram modelos *continuous bag of words (CBOW)* e *skip-gram*, popularmente conhecido como Modelos *Word2Vec.* A arquitetura CBOW tenta prever a palavra atual dada a anterior e a próxima, descartando a ordem das palavras, em uma janela de contexto fixo [Le and Mikolov, 2014].

O modelo *skip-gram* prevê as palavras ao redor, dadas a palavra atual. Esses modelos têm complexidade de treinamento robusta, e, portanto, podem ser usados para treinamento em grandes volumes de dados. Os vetores gerados por esses modelos mostraram identificar relacionamentos semânticos sutis entre as palavras [Le and Mikolov, 2014].

Figura 19 – Arquiteturas *CBOW* e *Skip-gram*




Traduzido de [Mikolov et al., 2013]

A figura 19 demonstra os dois modelos de arquitetura sendo que a *CBOW* prevê a palavra atual com base no contexto, e o *Skip-grama* prevê palavras ao redor, dada a palavra atual. Sendo P = Palavra e i = Índice da palavra dentro do texto.

### 2.3.8 Seleção de características

Dentro dos problemas enfrentados na categorização dos *tickets*, um problema relevante é a alta dimensionalidade de características, que são compostas por todos os termos únicos (palavras) que aparecem no texto do *ticket*. Nesse contexto e trabalhando com grandes volumes de *tickets* de entrada, ocorre de haver características redundantes ou irrelevantes, podendo atingir a casa de milhares de características, o que pode dificultar, ou até impedir, o correto treinamento dos modelos de Aprendizado de Máquina [Kadar et al., 2011].

Se faz vital atuar na redução do volume de característica e reduzir o custo computacional com processamento e armazenamento dessas informações. Nessa redução, se faz necessário escolher um subconjunto de características que melhor representem a informação contida no conjunto original, evitando prejudicar o trabalho de generalização e precisão de categorização dos modelos [Jan et al., 2013].

#### 2.3.8.1 Frequência no documento - FD

É um método estatístico que mede a frequência que um termo t ocorre em uma classe c, em particular. É definido pela equação (2.2).

Frequência no documento é uma medida estatística que tem o objetivo indicar a importância de uma palavra de um documento em relação a uma coleção de documentos ou em um corpus linguístico. Ela é frequentemente utilizada como fator de ponderação na recuperação de informações e na mineração de dados. Executa a medição da frequência que um termo $t$ ocorre em uma classe $c$, em particular. É definido pela equação:

$$FD(t,c) = \frac{C(t,c)}{\sum t_i \in T(c) C(t_i,c)}$$

Onde:

- O numerador é a quantidade de ocorrências de $t$ nos documentos da classe $c$;
- O denominador é o somatório das quantidades de ocorrências de todos os termos nos documentos da classe $c$;

### 2.3.9 Redução de dimensionalidade

Para se construir um modelo de classificação para o *tickets*, um dos desafios mais relevantes é a correta identificação de um conjunto representativo de características. Em modelos preditivos, numerosas características podem sobrecarregar o classificador, nesse contexto se faz necessário a execução da redução de dimensionalidade.

Consta na literatura técnicas de redução de dimensionalidade:

Tabela 4 – Quadro de técnicas de redução de dimensionalidade

| Trabalho | Conclusão |
|---|---|
| [Pearson, 1901] | Análise de componentes principais (PCA, do inglês Principal *Component Analysis*): utiliza a ortogonalizadão de vetores para converter um conjunto de variáveis possivelmente correlacionadas num conjunto de variáveis linearmente não correlacionadas (chamadas de componentes principais). |
| [Liu and Setiono, 1996] | Consistência de subconjunto (LVF, do inglês *Las Vegas Filter*): baseia-se em um algoritmo de busca aleatório que demonstra alta robustez (em termos de atributos redundantes ou correlações de ordem alta) quando utilizado com conjuntos de dimensionalidade alta. |
| [Hall, 2000] | Correlação de subconjunto (CFS, do inglês *Correlation-based Feature Selection*): baseia-se na hipótese central de que bons conjuntos de características possuem características altamente correlacionadas com a classe, ainda que não correlacionadas entre si. |
| [Guyon et al., 2002] | Máquinas de vetores de suporte (SVM-RFE, do inglês *Support Vector Machine Recursive Feature Elimination*): faz uso de *SVMs* com eliminação recursiva de características para efetuar redução de dimensionalidade, descartando atributos redundantes e produzindo subconjuntos compactos e, ao mesmo tempo, com grande capacidade de predição. |
| [Covões and Hruschka, 2011] | Filtro de silhueta simplificada (SSF, do inglês *Simplified Silhouette Filter*): consiste em agrupar atributos em subconjuntos de cardinalidade estimada automaticamente, podendo considerar correlações entre características e classes. |

Fonte: Elaborada pelo Autor

### 2.3.10 Algoritmos e modelos de classificação

Para se construir um modelo de classificação de texto a serem utilizados nas fases de treinamento e classificação, são utilizados algoritmos e modelos que terão como entrada os textos já preparados nas atividades de Pré-Processamento [Kadar et al., 2011, Venkataraman, 2016, Eckstein et al., 2016, Parmar et al., 2018].

#### 2.3.10.1 Naive Bayes

Os classificadores ingênuos de Bayes pertencem à um grupo de classificadores probabilísticos simples, baseados na utilização do teorema de Bayes com a suposição **ingênua** de independência entre os recursos. Foi incorporado ao quadro bayesiano três modelos: o modelo de Bernoulli, o modelo multinomial e o modelo de Poisson, resultando nos classificadores ingênuo Bayes (BNB), ingênuo multinomial Bayes (MNB) e Poisson ingênuo Bayes (PNB). Naïve Bayes é estudado desde a década de 1960 e originalmente foi utilizado

na recuperação de textos e continua sendo utilizado para categorização de textos [Tang et al., 2016].

#### 2.3.10.2 *Support Vector Machine - SVM*

*Support Vector Machine* ou Máquina de Vetores de Suporte é um método de aprendizado supervisionado utilizado para análise de dados e reconhecimento de padrões, utilizado para classificação e análise de regressão. Busca-se encontrar a maior margem para separar diferentes classes de dados, toma-se como entrada um conjunto de dados e prediz, para cada entrada dada, qual de duas possíveis classes a entrada faz parte. O SVM busca encontrar uma linha de separação, denominada **hiperplano** entre dados de duas classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes [Shafiabady et al., 2016].

# 3 Trabalhos relacionados

Foram realizadas buscas em 3 (três) bases de dados de artigos científicos: *Digital Bibliography & Library Project (DBLP)*, Portal de periódicos CAPES e *IEEE – Institute of Electrical and Electronic Engineers*. Foram buscados os artigos publicados entre Jan/2010 à Fev/2020 contendo simultaneamente os termos *Machine Learning*, *Text Classification* e *Ticket*. Nessa busca foram retornados 14 (quatorze) publicações.

Tabela 5 – Bases pesquisadas

| Bases | Strings de Busca | Período | Publicações |
|---|---|---|---|
| *DBLP* | "Machine Learning" | Jan/2010 | 3 |
| *CAPES* | + "Text Classification" | à | 3 |
| *IEEE* | + "Ticket" | Fev/2020 | 8 |

Fonte: Elaborada pelo Autor

Todas as publicações foram estudadas integralmente, identificando as técnicas utilizadas e os resultados obtidos na literatura para classificação e recuperação de *tickets* em língua estrangeira.

A Tabela 6 demonstra de forma resumida, o trabalho científico, a quantidade de *tickets* avaliados por cada trabalho, as técnicas utilizadas e as conclusões a que os autores chegaram com seus experimentos.

Tabela 6 – Quadro comparativo de trabalhos relacionados

| Trabalho | Tickets | Técnicas | Acurácia | Conclusão |
|---|---|---|---|---|
| Kadar2011 | 2.634 | Vector Space Model (VSM) e Multinomial Logistic Regression Classifier | 89,20% | A configuração supervisionada de aprendizado de máquina produziu resultados satisfatórios, mas exigiu grandes quantidades de dados para treinamento. O Aprendizado semi-supervisionado, com um custo moderado (poucas palavras rotuladas 200 palavras) resultou em desempenho satisfatório (74,20%) de acurácia. |
| Jan2013 | 29.212 | Conditional Random Fields (CRFs) e maximum Entropy Interpretation | 93,12% | A Origem dos *tickets* são chaves para vincular os *tickets* a vários recursos de outras unidades de negócios na entrega de serviços de TI. Sem a correta classificação, o conhecimento de diferentes conteúdos e domínios permanecem separados e desconectados. |
| Li2014 | 1.445 | Hierarchical Clustering, Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) | 91,24% | Os volumes de *tickets* podem exibir padrões complexos que não podem ser adequadamente modelados como função linear junto à ajustes sazonais. A Automatização é uma tarefa desafiadora para seleção dos métodos estatísticos para atender a variedade e volatilidades do grande volume de *tickets*. |

Continua na próxima página

**Tabela 6 – Continuação da página anterior**

| Trabalho | Tickets | Técnicas | Acurácia | Conclusão |
|---|---|---|---|---|
| Venkataraman2016 | 90.212 | Naive Bayes, Support Vector Machine e Multi-Layer Perceptron | 80,00% | A classificação do texto na vida real é uma questão desafiadora. Requer uma compreensão profunda dos dados e do tipo de algoritmos de limpeza de dados necessários para lidar com esses tipos de dados. Requer uma estreita coordenação com as diferentes fontes de dados para entender melhor os dados, para que as técnicas de dados apropriadas possam ser listadas, aplicadas e o resultado verificado. |
| Eckstein-2016 | 234 | Naive Bayes, C4.5, SVM, kNN | 45,00% | Os resultados mostram que a abordagem proposta é suficiente para obter informações de forma automatizada e escalável, após um treinamento inicial. A quantidade de registros utilizados era pequeno. Os autores acreditam que a aplicação de dados e técnicas de mineração de texto serão um meio eficaz de apoiar gerentes de marketing e inovação, especificamente, para criar relações de serviço duradouras e ofertas individualizadas. |
| Paramesh-2018 | 10.742 | Logistic regression (LR), K-Nearest-Neighbour(KNN), Multinomial Naïve Bayes (MNB) e Support vector machines (SVM) | 81,11% | Foi desenvolvido um classificador de *tickets* modelo para um dos serviços de infraestrutura de TI em tempo real. Foram utilizados métodos de classificação de conjuntos, juntamente com modelos de classificadores básicos para construir os classificadores. O desempenho de todos os classificadores de conjunto superou bem em comparação com os classificadores de base. |
| Parmar2018 | 5.500 | K-Nearest-Neighbour(KNN), Multinomial Naïve Bayes (MNB), Support vector machines (SVM), Decision Tree, Random Forest | 63,02% | O Trabalho contém uma análise comparativa dos resultados de cinco classificadores para classificar os problemas dos clientes em categorias predefinidas. O modelo foi aplicado em dados reais do cliente. Verificou-se que (SVM) apresentau o melhor desempenho entre os cinco classificadores. A precisão SVM é de 63,02% para o conjunto de dados fornecido. O modelo poderá ser utilizado para os engenheiros de aplicativos de suporte classificarem os defeitos automaticamente. Esse processo melhorará a eficiência e a precisão do processo de classificação de defeitos. |
| Silva2018 | 10.000 | K-Nearest-Neighbour(KNN), Long Short Term Memory (LSTM) | 89,00% | A categorização manual de *tickets* é um processo que leva tempo, que pode ser reduzido com automação da categorização. Na maioria das vezes os *tickets* são atribuídos a uma categoria incorreta. Com isso é possível concluir quais são as categorias críticas e obter uma melhor desempenho na categorização. Os autores acreditam que reduzindo os erros de categorização de *tickets*. Isso é determinante para obter uma correta atribuição e reduzir o tempo perdido e melhorar toda a rota do *ticket*. |
| Lyubinets-2018 | 58.871 | Naive Bayes, Support vector machines (SVM), Term Frequency Inverse Document Frequency (TF-IDF) | 88,20% | As abordagens que usam RNNs nos dados de incorporação de palavras superam as soluções clássicas, que abre portas para muitos casos de uso prático. Todavia, as precisões resultantes ainda estão poderão obter um melhor desempenho, levando em consideração que as diversas classes e a classificação do texto continua sendo um problema em aberto. Naive Bayes não é uma boa solução para a classificação de várias classes. Soluções já antigas como TF-IDF com SVM ainda mostram bons resultados e pode ser uma ótima solução para casos onde os recursos são limitados. |

Continua na próxima página

**Tabela 6 – Continuação da página anterior**

| Trabalho | Tickets | Técnicas | Acurácia | Conclusão |
|---|---|---|---|---|
| Qamili-2018 | 445.000 | Naive Bayes, Support vector machines (SVM), Logistic Regression, Decision Trees, Random Forest | 86,10% | O objetivo geral deste estudo foi projetar um sistema eficiente para fornecer *tickets* totalmente automatizados. Foram abordadas 3(três) dimensões diferentes: spam filtragem, atribuição automática de *tickets* e análise de sentimentos. Foi implementado um aplicativo de protótipo para um cenário real, com foco principal na detecção de *tickets* de spam, e como segundo passo, atribuição automática de *tickets* genuínos para o departamento correspondente da empresa. Na detecção automática de spam, foram aplicados combinação de vários modelos para abordar as falsas questão positiva. O modelo híbrido filtrou estritamente os falsos positivos, mas em troca de permitir que mais *tickets* de spam sejam classificados como bilhetes legítimos. Devido à falta de anotação do conjunto de dados para a análise de sentimentos, não foi possível treinar os classificadores. |
| Phetrung-napha-2019 | 4.400 | Bernoulli Naive Bayes, Guassian Naive Bayes, Decision Tree, K-Nearest-Neighbour(KNN), Linear SVC, Logistic Regression | 80,63% | Este trabalho investigou algoritmos de aprendizado de máquina para classificar relatórios de bugs e solicitações de recursos que apareceu nas análises de usuários de aplicativos para dispositivos móveis na App Store e Play Store. Isso poderia facilitar a geração automatizada de *tickets* em um quadro de Jira de um projeto de software. Várias técnicas de PNL foram aplicadas para processar comentários de revisão textual, determinar revisões duplicadas com base na semelhança semântica e resumir o conteúdo das revisões. |
| Kallis-2019 | 30.000 | fastText linear classifier, Vectorial Representation | 83,10% | Neste trabalho, foi desenvolvido um aplicativo denominado *Ticket Tagger* que automaticamente atribui rótulos aos *tickets* abertos nos projetos do GitHub. Segundo os autores é possível que outros desenvolvedores que desejarem de melhorar o processo de manutenção de problemas por meio do classificação automatizada possam integrar facilmente com o *Ticket Tagger*. O núcleo do *Ticket Tagger* é representado por um modelo de *Machine Learning* que analisa o título e a descrição textual do *ticket*, a fim de determinar se tal questão pode ser rotulada como um relatório de bug, uma solicitação de recurso ou uma pergunta. |
| Li2019 | 14.651 | Naive Bayesian, TextRank, TF-IDF, Word2vec | 77,10% | Neste trabalho foi apresentado um método estruturado de vetor de espaço para construir um Classificador de Bayes para classificar automaticamente *tickets* de reclamações ferroviárias. Como o texto de reclamação ferroviária tem as características de negócios amplos, eventos diversos, registro textual grave, interferência séria e informações inúteis, o método tradicional de classificação de texto é adotado diretamente e a precisão da classificação de texto é baixa. A precisão da classificação é de apenas 0,521 pelo texto original. Portanto, este artigo adota o método de extração de valor de recurso para melhorar ainda mais a precisão da classificação do texto. Neste trabalho, três algoritmos de extração de recursos são comparados, incluindo TF-IDF, TextRank e Word2vec. |

Continua na próxima página

**Tabela 6 – Continuação da página anterior**

| Trabalho | Tickets | Técnicas | Acurácia | Conclusão |
|---|---|---|---|---|
| Kurz2020 | 20.837 | Bidirectional Encoder Representations from Transformers (BERT), Bi-Encoder, Cross-Encoder | 83,98% | Neste trabalho os autores abordam a questão de saber se os resultados alcançado com modelos de redes neurais aplicadas em grandes enciclopédicas baseadas na web podem ser transferidos para vinculação de *tickets* em um contexto de negócios. A Tarefa realizada foi de classificação de *tickets* em um cenário de aplicação de engenharia mecânica: Esses *tickets* referem-se a peças da máquina, como "Flansch" / "Luftanschluss" (flange) ou sintomas de erro como "Leck" / "O¨ laustritt" (vazamento). Portanto foram vinculados à documentos na base de conhecimento de tíquetes coletados por técnicos de serviço no campo. Os autores concluem que que um conjunto de transformações simbólicas e uma abordagem neural usando BERT atinge resultados impressionantes tanto em trechos da Wikipedia quanto em um mundo real com um conjunto de dados de um empresa industrial. |

Fonte: Elaborada pelo Autor

Após a identificação e avaliação das técnicas de Aprendizado de Máquina utilizadas por esses trabalhos, para classificação e recuperação de *tickets* em língua estrangeira foram selecionadas as 4 (quatro) técnicas que apresentaram os melhores resultados. Essas técnicas estão relacionadas na tabela 7 e são utilizadas para construção do experimento nessa pesquisa.

Tabela 7 – Algoritmos utilizados na literatura

| Algoritmo |
|---|
| Naive Bayes |
| SVM - Support Vector Machine |
| Long Short Term Memory (LSTM) |
| Bidirectional Encoder Representations from Transformers (BERT) |

Fonte: Elaborada pelo Autor

# 4 Metodologia

## 4.1 Tipo de pesquisa

Para se alcançar os objetivos propostos neste projeto, propõe-se as seguintes abordagens de pesquisas segundo a classificação proposta por [Gil, 2007] e [Córdova and Silveira, 2009].

Do ponto de vista da natureza da pesquisa:

- **Pesquisa Aplicada:** pois tem o objetivo de gerar conhecimento para a aplicação prática, dirigidos à solução de problemas específicos.

Do ponto de vista da abordagem ao problema

- **Pesquisa Quantitativa:** pois pretende-se aplicar métodos matemáticos e estatísticos para testar e estabelecer de forma quantitativa relacionamentos entre variáveis respostas e explicativas, buscando-se avaliar quais os modelos quanto à acurácia do ponto de vista estatístico.

Do ponto de vista dos objetivos

- **Pesquisa Explicativa:** pois tem por objetivo identificar os modelos pela taxa de recuperação e identificação de *ticket* duplicados.

Do ponto de vista das técnicas de pesquisas

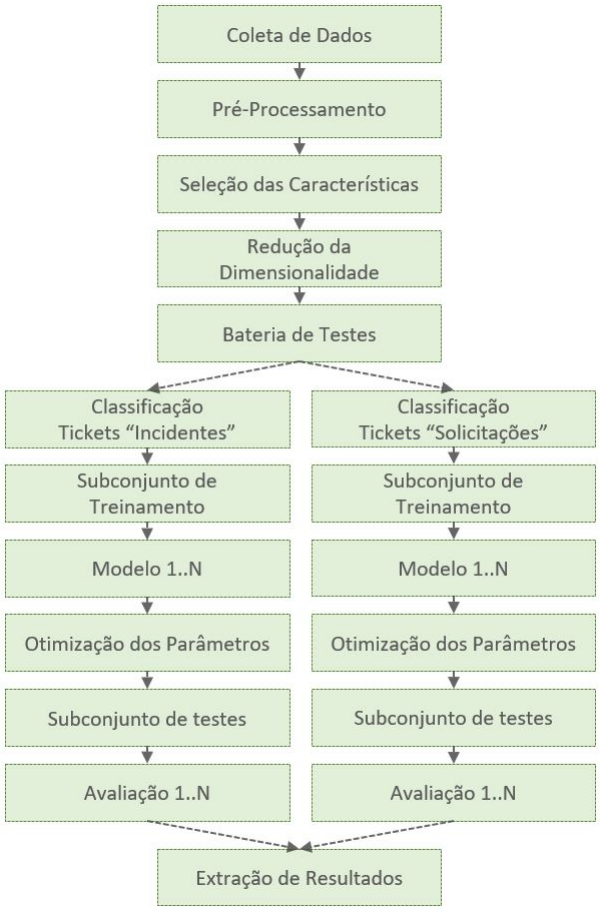
- **Pesquisa Experimental:** pois pretende testar modelos de identificação de *tickets* em dados reais buscando ajustá-los de acordo com os dados e identificar e avaliar o resultado de acurácia de acordo com critérios objetivos.

- **Procedimentos Metodológicos:** Conforme narrado, iniciou-se por uma revisão de literatura realizadas em 3(três) bases de dados: Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), *DBLP (Digital Bibliography and Library Project)* e *IEEE Xplore digital library*; Foram encontrados 14(Quatorze) trabalhos sobre classificação automática de *tickets*, utilizando as combinações das strings de busca: "Machine Learning", "Text Classification", "*Ticket*". Foram analisadas e escolhidas as publicações que estão em consonância com a pesquisa proposta.

## 4.2 Modelo proposto

Para se avaliar a acurácia dos algoritmos de Aprendizado de Máquina na identificação de *tickets* duplicados escritos na língua portuguesa foi elaborado o modelo ilustrado no diagrama da figura 20, cujos elementos são explicados individualmente na sequência.

Figura 20 – Diagrama de blocos da metodologia proposta




Fonte: Elaborada pelo Autor

### 4.2.1 Coleta de dados

A Coleta de dados, iniciou-se pela busca de empresas brasileiras de prestação de serviços de tecnologia, que tenham sistemas de gerenciamento de solicitações de usuários. Considerou-se importante que tal empresa realizasse suas atividades em todo o território nacional e contasse com um número relevante de usuários e tickets em sua base de dados. Para esse trabalho, também, era condição vital que os *tickets* registrados já estivessem identificados quanto à sua possível duplicação com outro(s) *ticket(s)* já existentes na base de dados, portanto seria necessário um base de dados real de produção.

Além dos desafios inerentes a disponibilização dessas informações por parte das empresas, somou-se a esse desafio a regulamentação trazida pela Lei Geral de Proteção de Dados Pessoais (LGPD ou LGPDP), que é legislação brasileira que regula as atividades de tratamento de dados pessoais. Dessa forma, todos os dados sensíveis de identificação dos usuários e empresas clientes não fazem parte dos dados analisados por esse trabalho. Todavia essa exclusão de dados não trouxe impacto ao experimento, uma vez que são utilizados os dados dos campos resumo e descrição dos *tickets* para a recuperação e classificação pelas técnicas de Aprendizado de Máquina.

A Base de dados que foi utilizada por esse trabalho pertence a empresa brasileira Microcity, trata-se de uma empresa terceirizadora de ativos e serviços para infraestrutura de TI (Locação de equipamentos e sistemas). Atualmente a Microcity é a maior empresa de terceirização de lans e desktops do Brasil com 21,1% de participação de mercado, base instalada de 300 mil equipamentos e mais de 130 mil atendimentos (*tickets*) com *Service Level Agreement* superior a 97%. A Empresa possui uma carteira de 150 clientes ativos, dentre eles grandes grupos de drogarias, planos de saúde, redes de supermercados varejistas e fábrica de brinquedos.

A Base de dados disponibilizada é composta de 132.703 tickets contendo registros de incidentes, solicitações de informações, serviços ou equipamentos durante o período de 1(um) ano. As informações contidas nos tickets foram disponibilizadas com o texto original do usuário solicitante, sem nenhum tipo de tratamento manual e/ou automático por parte do sistema de gerenciamento.

Todos os *tickets* desse experimento, já foram anteriormente atendidos pela Central de Serviços que efetuou a classificação manual, bem como também se encontram com status *encerrados* com suas respectivas soluções registradas.

### 4.2.2 *Dataset*

Um conjunto de dados ou do inglês *dataset* é uma coleção de dados normalmente tabulados. Cada elemento (ou indivíduo) pode representar diversas características. Cada coluna representa uma variável em particular. Cada linha corresponde a um determinado membro do conjunto de dados em questão. Cada valor é conhecido como um dado. O conjunto de dados pode incluir dados para um ou mais membros, correspondente ao número de linhas [Zighed et al., 2000].

Na base de dados utilizado por esse experimento, temos um *dataset* composto de 10 (dez) campos de dados referentes ao *ticket* em questão. Na tabela 8 é demonstrado o exemplo de um *ticket* do *dataset*.

Tabela 8 – Conjunto de dados - exemplo de 1(um) *ticket*

| Nome do Campo | Conteúdo do Campo |
|---|---|
| Chamado | IN252455 |
| Tipo | Solicitação |
| Meio de abertura | Telefone |
| Data de abertura | 23/03/2020 07:27 |
| Data de fechamento | 23/03/2020 12:02 |
| Resumo | NOBREAK NÃO SEGURA CARGA |
| Descrição | Usuário informa que nobreak não está segurando carga. Gentileza verificar. |
| Solução | Solução do atendimento: Realizado a substituição do nobreak. Testes para validar a solução: Realizado os testes no nobreak, segurou carga e passou energia normalmente para os equipamentos. Horário do atendimento (Início e Fim): 11:40 às 12:00 (horário local). Fechamento validado pela central com o usuário: XPTO Equipamento com Avaria: N/A Validação do Autômatos: N/A Validação do Hardware (HD, processador e memória): N/A Preenchimento e assinatura do RAT: PRIME |
| Duplicado | Não |
| Id. Duplicado | 0 |

Fonte: Elaborada pelo Autor

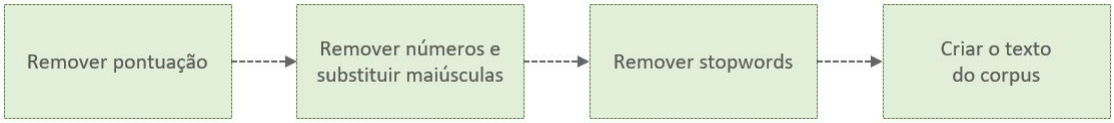
As informações contidas no *dataset* de *tickets* são utilizados por este trabalho da seguinte forma:

- **Id. Ticket** Este campo é o identificador único do *ticket*, quando um outro *ticket* é considerado duplicado, é anotado no *ticket* duplicado o *identificador único* do *ticket* de referência.
- **Tipo** Este campo é primeiro agrupador e dado rotulado do *ticket*, informa em qual grupo o *ticket* se encontra. Para este campo, temos os seguintes valores possíveis (*Incidente*, *Pergunta*, *Solicitação*).
- **Meio de Abertura** Este campo informa o meio pelo qual o *ticket* foi registrado no repositório.
- **Data/Hora abertura e Data/Hora fechamento** São respectivamente as informações de Data/Hora da abertura do *ticket* pelo usuário e do fechamento do *ticket* pela central de serviços, a subtração da maior pela menor indica o período de tempo em que o *ticket* permaneceu em atendimento, ou o tempo total utilizado para a solução do mesmo.
- **Título** Campo informado pelo usuário, com uma breve descrição do *ticket*, é utilizado como base de conhecimento para os classificadores.
- **Descrição** Campo no qual o usuário deve descrever de forma ampla sua necessidade no *ticket*.
- **Solução** Campo que os o(s) agente(s) de atendimento utilizam para descreverem a solução adotada para a solução de cada *ticket*.
- **Duplicado** Campo que os o(s) agente(s) de atendimento utilizam para informar se o *ticket* é um duplicata de um outro *ticket* já armazenado no banco de dados. Esse verificação foi realizada nesse *dataset* de forma manual pela central de serviços.
- **Id. Duplicado** Campo que o(s) agente(s) de atendimento utilizam para informar o *identificador único* do *ticket* de referência quando o mesmo é considerado uma duplicada, caso contrário, esse campo recebe o valor 0 (zero).

### 4.2.3 Pré-processamento

Após a disponibilização da base de dados, a primeira atividade realizada foi o tratamento das informações contidas no *ticket* pelos métodos de pré-processamento descritos anteriormente.

Figura 21 – Sequência de atividades de pré-processamento




Fonte: Elaborada pelo Autor

### 4.2.4 Seleção de características

Nessa atividade, foram avaliados os textos de todos os *tickets* do conjunto de dados e extraídos suas características em conformidade com os métodos descritos na seção 2.3.8, gerando para cada texto de *ticket* uma representação numérica, que foi utilizada pelos algoritmos de Aprendizado de Máquina.

### 4.2.5 Redução da dimensionalidade

Nessa atividade, foi aplicado o método descrito na seção 2.3.9 para reduzir a dimensionalidade do conjunto de dados gerados na etapa anterior, foi preservado as características significativas de cada *ticket* e descartadas as características irrelevantes.

selecionadas as 4 (quatro) técnicas que apresentaram os melhores resultados. Essastécnicas estão relacionadas na tabela 7 e são utilizadas para construção do experimento nesta pesquisa.

### 4.2.6 Algoritmos utilizados

Para se atingir os objetivos deste trabalho, conforme descrito no capítulo 3 foram selecionadas as 4 (quatro) técnicas de *Machine Learning* que apresentaram os melhores resultados na classificação e recuperação de Solicitação de Usuário em língua estrangeira. as técnicas utilizadas pelo experimento foram:

- **Long Short Term Memory (LSTM)** - Descrito na seção: 2.2.2.3
- **Bidirectional Encoder Representations from Transformers (BERT)** - Descrito na seção: 2.2.2.4
- **Naive Bayes** - Descrito na seção: 2.3.10.1

- **SVM - Support Vector Machine** - Descrito na seção: 2.3.10.2

os dados foram submetidos ao processamento dos algoritmos, e as atividades abaixo descritas foram executadas para cada algoritmos individualmente.

### 4.2.7 Bateria de testes

Nessa atividade, o conjunto total de dados, foi aleatoriamente dividido em dois subconjuntos, sendo que primeiro subconjunto foi utilizado para treinamento do modelo contendo 70% dos registros, e o segundo subconjunto foi utilizado para a execução da classificação contendo 30% dos registros.

### 4.2.8 Modelo

Nessa atividade, foi gerado um modelo resultado da atividade de treinamento do algoritmo de aprendizado. Cada modelo resultante foi persistido no sistema de arquivos para uso posterior.

### 4.2.9 Otimização de parâmetros

Nessa atividade, durante o treinamento do modelo, foram selecionados e ajustados os parâmetros dos algoritmos com o objetivo de gerar melhores resultados.

### 4.2.10 Avaliação

Nessa atividade, cada modelo gerado é avaliado por meio da classificação do subconjunto de testes, e o resultado de cada métrica foi calculado.

### 4.2.11 Extração de resultados

Nessa atividade, após a avaliação dos modelos, foram considerados e avaliadas as pontuações de escores F1, precisão e revocação. A média da pontuação de cada um dos modelos foi comparada.

## 4.3 Métrica de desempenho

Métricas de avaliação de desempenho são essenciais para descobrir os resultados efetivos que o modelo alcançou. No contexto de *tickets*, o objetivo é avaliar se um *ticket* do conjunto é uma réplica de outro *ticket* no mesmo conjunto de dados. Para este projeto, foi utilizada a métrica de Acurácia também denominada *taxa de sucesso*, como métrica de avaliação. Para no caso binário, a métrica de precisão é definida como:

$$Acurácia = \frac{tp + tn}{(tp + fp + tn + fn)}$$

[Foody, 2002, p. 187]

Onde *tp* são os verdadeiros positivos *true positives*, *fp* os falsos positivos *false positives*, *tn* verdadeiros negativos *true negatives*, e *fn* os falsos negativos *false negatives*. A Precisão refere-se à porcentagem de previsões corretas feitas pelo modelo quando comparado com as classificações reais e pode ser calculado como a soma das classificações corretas (a soma dos elementos em diagonal principal da matriz de confusão) dividida pela número total de classificações (a soma de todos os elementos de a matriz de confusão) [Li and Li, 2019].

# 5 Experimentos

## 5.1 Técnicas de aprendizado de máquina

A fim de consultar o *dataset* e estabelecer um padrão ouro para comparar os resultados, foi realizado um experimento para classificação e recuperação de *tickets*. As seguintes técnicas foram empregadas: LSTM, BERT, Naive Bayes e SVM. As técnicas utilizadas foram totalmente implementadas com o uso da linguagem Python.

As implementações de Naive Bayes e SVM foram baseadas em [Phetrungnapha and Senivongse, 2019], foi utilizado a biblioteca sklearn [Buitinck et al., 2013], e para gerar uma representação da descrição do texto foi aplicado o modelo TFIDF.

A implementação de LSTM foi baseada em [Lyubinets et al., 2018], foi utilizado a biblioteca TensorFlow com a api Keras [Geron, 2017].

Para a implementação de BERT as descrições do *ticket* foram representadas como um vetor de 1024 posições usando BERT multilingual [Xiao et al., 2019] baseado em BERT [Devlin et al., 2018].

Nessas técnicas, a similaridade de cosseno foi usada para comparação de sentenças.

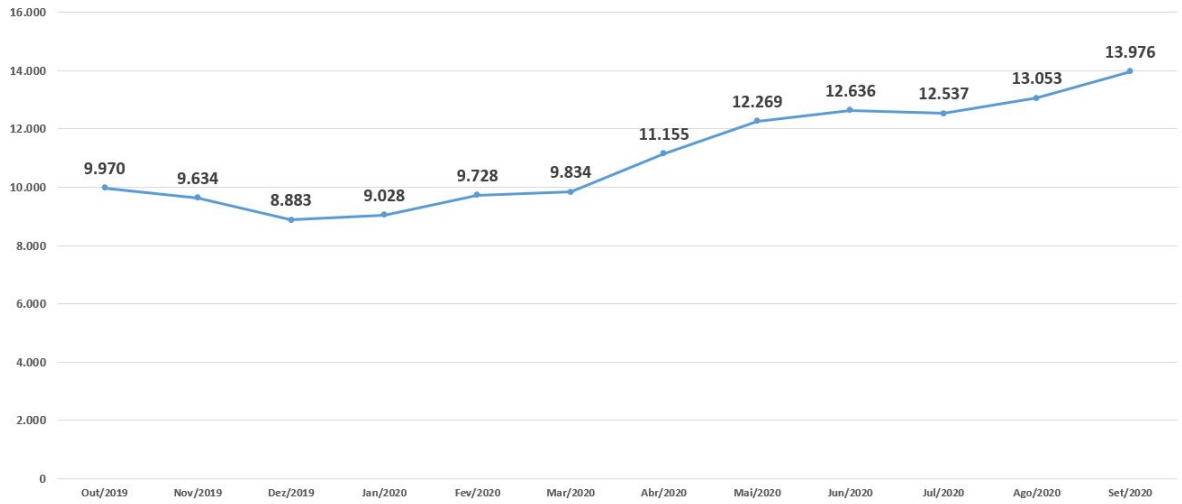
## 5.2 Análise estatística do *dataset*

Conforme descrito, foi utilizado por esse experimento uma base de dados única, composta de 132.703 *ticket* contendo registros de *Incidentes/Problemas*, *Perguntas/Dúvidas* e *Solicitações de Tarefas/Serviços*. Registrados no período compreendido entre Out/2019 e Set/2020. Trata-se de uma base de dados real de produção de uma empresa brasileira de tecnologia de informação. São descritas a seguir análises das informações dessa base de dados.

#### 5.2.0.1 Período de abertura dos tickets

Referente ao período de abertura dos *tickets* foi constatado que houve um crescimento acelerado do número de ocorrências a partir do mês de março/2020. Este aumento ocorreu sobretudo devido a pandemia do novo Coronavírus que obrigou os usuários da organização a mudarem seu local físico de trabalho para o trabalho em casa, e com essa mudança, sua interação para com a organização passou a ser exclusivamente digital, sendo que todas as ocorrências de *Incidentes/Problemas*, *Perguntas/Dúvidas* e *Solicitações de Tarefas/Serviços* obrigatoriamente foram registradas via *tickets* aumentando ainda mais

Figura 22 – Distribuição dos tickets data de abertura

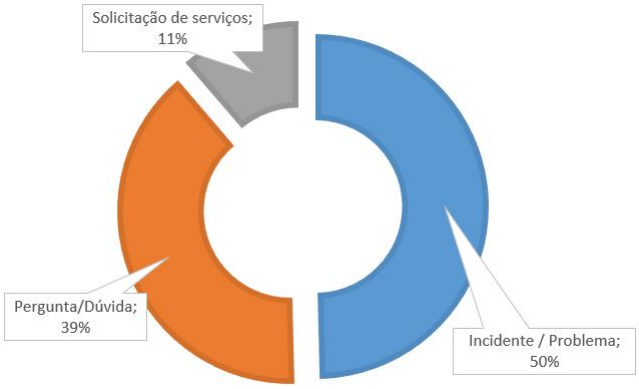



Fonte: Elaborada pelo Autor

a relevância de deste trabalho na busca por um modelo automatizado para detecção e recuperação de *tickets* duplicados.

#### 5.2.0.2 Tipos de solicitações dos *tickets*

Figura 23 – Distribuição dos tickets por tipo de solicitação

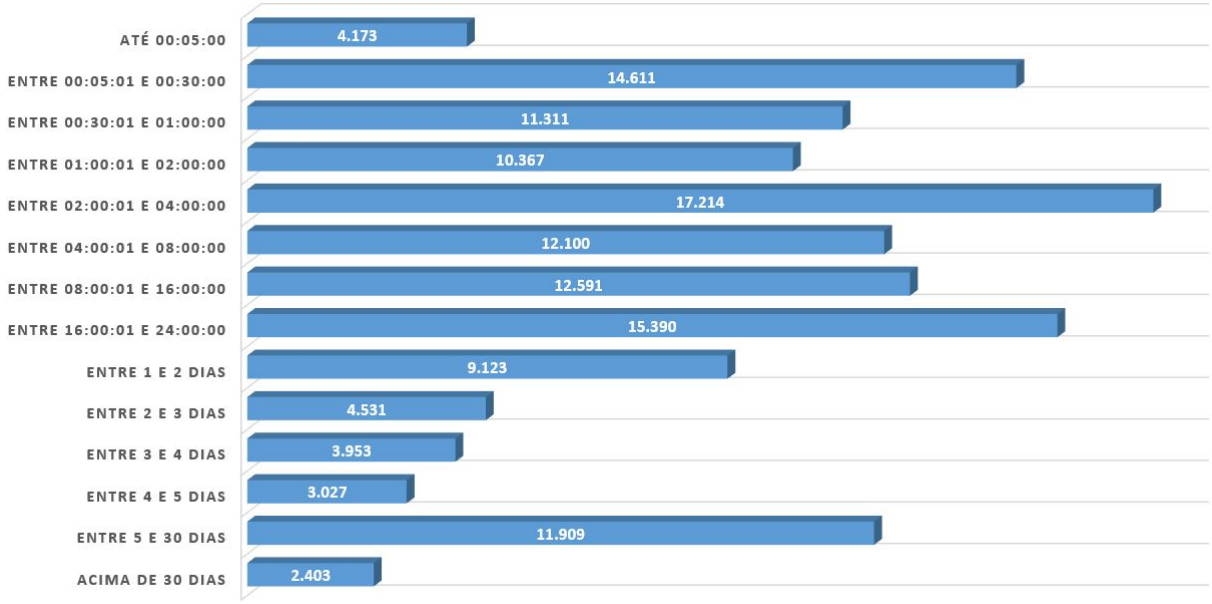



Fonte: Elaborada pelo Autor

Referente a distribuição dos *tickets* pelo Tipo da Solicitação, foi constatado que 50% (Cinquenta por cento) das ocorrências são de usuário com relatos de **Incidentes ou Problemas**. 11% (Onze por cento) são ocorrências de usuários que já identificaram de antemão a solução para sua demanda e na abertura do *tickets* já realização a correta **Solicitação de serviços** para lhes apoiar na ocorrência em questão e 39% (Trinta e nove por cento) são ocorrências de usuários que estão na categoria **Pergunta/Dúvida** estes usuários estão com dúvidas técnicas e/ou de negócios. Nosso trabalho tem um maior foco neste último grupo, uma vez que solucionado a Dúvida do usuário, o *ticket* poderá ser encerrado imediatamente.

#### 5.2.0.3 Tempo de atendimento dos tickets

Figura 24 – Distribuição dos tickets pelo tempo de atendimento




Fonte: Elaborada pelo Autor

O Tempo de atendimento do *tickets* é o tempo total utilizado na tratativa da demanda compreendido desde o momento da abertura do *tickets* até o seu encerramento (conclusão). Referente a este tempo, pudemos observar 4(Quatro) grandes grupos distribuídos da seguinte forma:

- **Tempo de atendimento de até 24 Horas (1 dia):** Este grupo conta com 97.757 *tickets* que correspondem à 73,66% do total de *tickets* analisados. Este grupo é composto prioritariamente com ocorrências de *Perguntas/Dúvidas*, *Solicitação de Serviços* de menor complexidade e *Incidentes/Problemas* também de menor complexidade.

- **Tempo de atendimento entre 1 e 5 dias:** Este grupo conta com 20.634 *tickets* que correspondem à 15,55% do total de *tickets* analisados. Este grupo é composto prioritariamente com ocorrências de *Perguntas/Dúvidas* de maior complexidade onde o demanda é repassada ao *Subject Matter Expert* (SME), também é composto de *Solicitação de Serviços* e *Incidentes/Problemas* ambos de complexidade média.

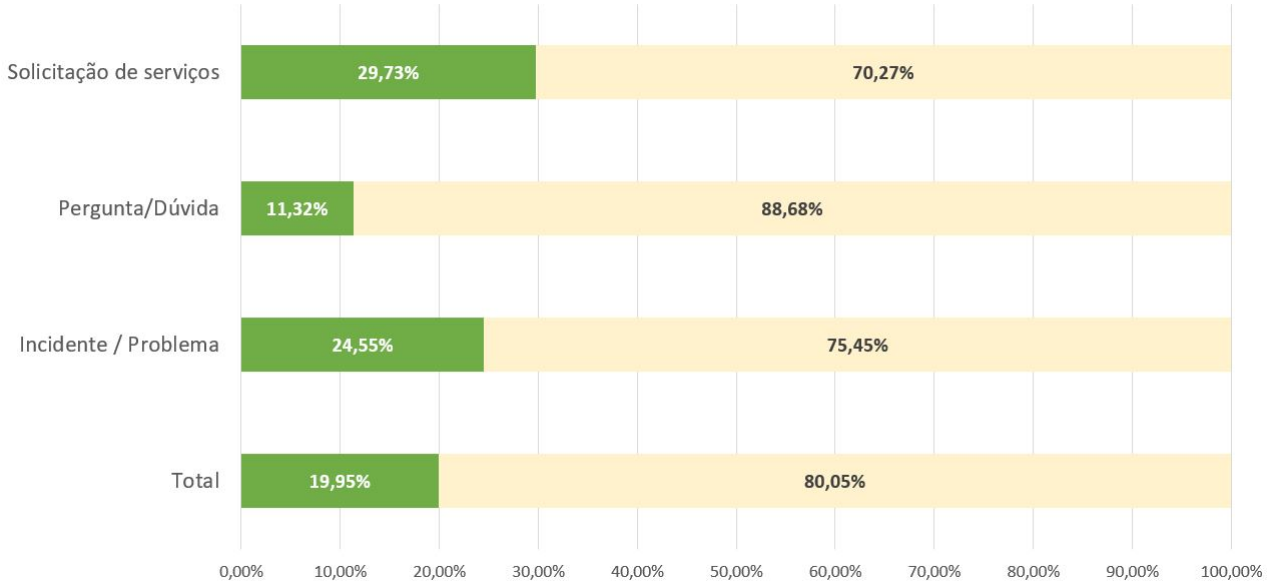
- **Tempo de atendimento entre 5 e 30 dias:** Este grupo conta com 11.909 *tickets* que correspondem à 8,74% do total de *tickets* analisados. Este grupo é composto prioritariamente com ocorrências de *Solicitação de Serviços* e *Incidentes/Problemas*

ambos de complexidade alta que necessitam de uma maior intervenção técnica por parte da organização.

- **Tempo de atendimento acima de 30 dias:** Este grupo conta com apenas 2.403 *tickets* que correspondem à 1,81% do total de *tickets* analisados. Este grupo é composto prioritariamente com ocorrências de *Solicitação de Serviços* e *Incidentes/Problemas* onde são é necessário o usuário aguardar a liberação de uma nova versão em casos de Softwares/Plataformas ou aguardar a Compra/Entrega de equipamentos de hardware.

#### 5.2.0.4 Distribuição dos *tickets* duplicados por classificação

Figura 25 – Distribuição dos tickets duplicados por classificação




Fonte: Elaborada pelo Autor

Evoluindo na avaliação referente aos *tickets* duplicados que são o objeto deste trabalho, pudemos observar que na base de dados foram anotados a partir da classificação manual dos técnicos da empresa (rotulação manual) 106.229 *tickets* duplicados, o que representa 80,05% do total de *tickets* existentes. Este percentual se mostrou bastante relevante e sugere que a cada 5 (cinco) novos *tickets* abertos, 4 (quatro) apresentam alta similaridade semântica (maior que 90,00 %) indicando se tratar de ocorrência duplicada.

Analisando a distribuição dos *tickets* duplicados por classificação, identificamos que este percentual é ainda maior quando a classificação é *Pergunta/Dúvida*, nessa classe temos o percentual de 88,68% de *tickets* anotados como duplicados. Este percentual indica que a cada 100 (cem) novos *tickets* abertos, 88 (Oitenta e oito) também apresentam alta similaridade semântica (maior que 90,00 %) indicando se tratar de ocorrência duplicada.

Na análise dos *tickets* da categoria *Pergunta/Dúvida*, foi constatado que uma vez respondido o questionamento do usuário de forma satisfatória, o *tickets* foi encerrado. Dessa forma, essa categoria se mostrou a mais indicada para a automação de solução das demandas com um modelo de Aprendizado de Máquina utilizando as técnicas discutidas neste trabalho.

## 5.3 Resultados

Nesta seção, são apresentados os resultados obtidos na realização do experimento conduzido nesta pesquisa e também é apresentado a discussão dos resultados obtidos.

### 5.3.1 Dados históricos

Os dados históricos dos atendimentos dos *ticket* (*dataset*) foram utilizados para o treinamento dos modelos de Aprendizado de Máquina avaliados por esse trabalho. A Quantidade de dados históricos (*ticket*) é um dos fatores de aumento do percentual de sucesso dos modelos de Aprendizado de Máquina na atividade de classificação e recuperação de *ticket* duplicados [Phetrungnapha and Senivongse, 2019].

Trabalhos anteriores sugerem que o treinamento dos modelos de Aprendizado de Máquina com grandes quantidade de dados reais eleva significativamente a acurácia dos modelos avaliados [Li and Li, 2019, Phetrungnapha and Senivongse, 2019, Parmar et al., 2018].

No capítulo 3 foram avaliados 14 (quatorze) publicações que utilizaram Aprendizado de Máquina para classificação e recuperação de *ticket* duplicados em língua estrangeira (inglês). Cada uma dessas publicações utilizou uma quantidade de registros para seu experimento, variando de 234 a 445.000 registros. O Experimento realizado nesse trabalho, utilizou um *Dataset* contendo 132.703 registros de *ticket*, representando portando um total de registros maior que o dobro da média de registros das publicações avaliadas.

### 5.3.2 Dados de treino e teste

Os dados utilizados nos experimentos de Aprendizado de Máquina normalmente são separados em duas partes, sendo uma parte para treinamento dos algoritmos e uma parte distinta dos dados para testes dos mesmos algoritmos. A Parte utilizada para treinamento normalmente corresponde à 70% do conjunto total de testes [Pikies and Ali, 2019, Geron, 2017, Paramesh et al., 2018]. Entretanto, outras publicações sugerem que um percentual maior de dados para testes aumenta a eficácia dos modelos de Aprendizado de Máquina para as atividades de classificação [Sanh et al., 2019, Liu et al., 2019, Devlin et al., 2018, Kurz et al., 2020].

Para a realização do experimento desse trabalho, foi considerado 4 (quatro) cenários distintos de treinamentos, conforme demonstrado na tabela 9.

Tabela 9 – Cenários de separação dos dados de treinamento e testes

| Cenário | Dados de Treinamento | Dados de Testes |
|---|---|---|
| Cenário A | 70 % dos registros | 30 % dos registros |
| Cenário B | 80 % dos registros | 20 % dos registros |
| Cenário C | 90 % dos registros | 10 % dos registros |
| Cenário D | 95 % dos registros | 5 % dos registros |

Fonte: Elaborada pelo Autor

Em cada um dos cenários, foram avaliados Naive Bayes [Phetrungnapha and Senivongse, 2019], SVM [Qamili et al., 2018], LSTM [Lyubinets et al., 2018] e BERT [Devlin et al., 2018]. Para uma comparação coerente, o mesmo conjunto de dados de treinamento foi utilizado para cada algoritmo.

### 5.3.3 Determinando *tickets* duplicados

São considerados duplicados, os *tickets* que os textos de suas descrições são semelhantes o suficiente entre si. Foi utilizado nesse experimento a análise de similaridade do texto semântico. Para essa análise é necessário determinar o percentual mínimo de similaridade (limiar) que as descrições dos textos necessitam ter.

Foi realizado um experimento para determinar o limiar de similaridade necessário para classificar um *ticket* como duplicado.

Para encontrar o limiar, foi determinado a pontuação semântica de similaridade de texto em pares de *tickets*. Foram analisados os pares já anotados manualmente como duplicatas. Foi realizado a vetorização desses textos e encontrado a distância entre os vetores. Em seguida, se qualquer par de *tickets* tivesse a pontuação de similaridade que não fosse menor do que o limite, o par foi considerado duplicado. Caso contrário, foi considerado como não duplicado. O desempenho de cada valor limite na detecção dos *tickets* duplicados foi avaliado em relação ao conjunto verdade.

A matriz de confusão e o desempenho de detecção para cada valor limite é apresentada na tabela 10. Neste contexto, a maior preocupação foi com o falso positivo, pois não é desejável considerar um *tickets* duplicado por sua pontuação de similaridade, quando na verdade, não é duplicado. O falso positivo deve ser mantido minimizado, enquanto a alta precisão é desejável. Neste caso, a quantidade de falso positivo diminuiu, mas a precisão

aumentou, enquanto o limite subiu para 0,78 antes que o desempenho se tornasse estável. Assim, 0,78 foi considerado ser o limite para a avaliação de duplicidade.

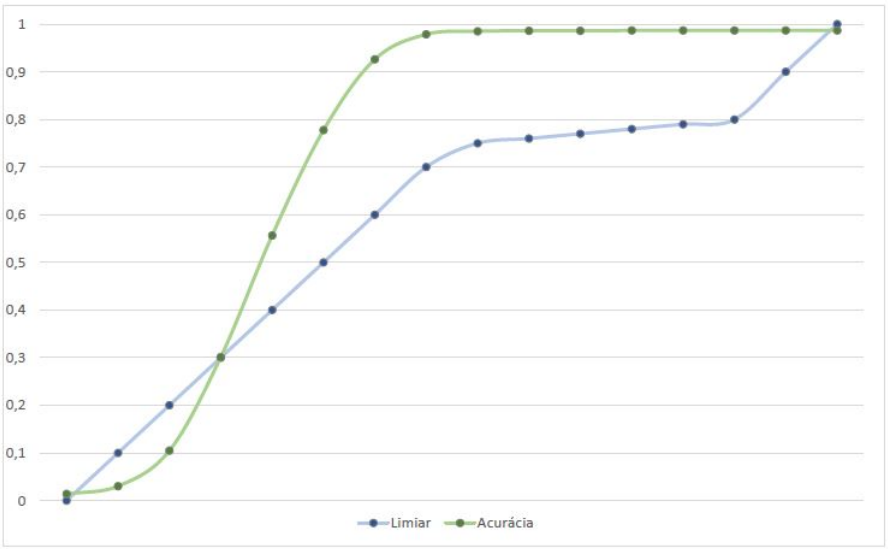
Tabela 10 – Desempenho de detecção de *tickets* duplicados pela variação do limiar

| Limiar | VP | VN | FP | FN | Acurácia |
|---|---|---|---|---|---|
| 0 | 57 | 1.413 | 104.760 | 0 | 0.0138 |
| 0,1 | 1.808 | 1.413 | 103.008 | 0 | 0,0303 |
| 0,2 | 9.719 | 1.413 | 95.098 | 0 | 0,1048 |
| 0,3 | 30.682 | 1.300 | 74.134 | 113 | 0,3011 |
| 0,4 | 58.087 | 1.017 | 46.729 | 396 | 0,5564 |
| 0,5 | 81.819 | 791 | 22.997 | 622 | 0,7776 |
| 0,6 | 97.979 | 396 | 6.837 | 1.017 | 0,9261 |
| 0,7 | 103.856 | 113 | 961 | 1.300 | 0,9787 |
| 0,75 | 104.647 | 0 | 170 | 1.413 | 0,9851 |
| 0,76 | 104.760 | 0 | 57 | 1.413 | 0,9862 |
| 0,77 | 104.760 | 0 | 57 | 1.413 | 0,9862 |
| **0,78** | **104.816** | **0** | **0** | **1.413** | **0,9867** |
| 0,79 | 104.816 | 0 | 0 | 1.413 | 0,9867 |
| 0,8 | 104.816 | 0 | 0 | 1.413 | 0,9867 |
| 0,9 | 104.816 | 0 | 0 | 1.413 | 0,9867 |
| 1 | 104.816 | 0 | 0 | 1.413 | 0,9867 |

Fonte: Elaborada pelo Autor

Onde *VP* são os verdadeiros positivos, *VN* os verdadeiros negativos, *FP* os falsos positivos e *FN* falsos negativos.

Figura 26 – Variabilidade da acurácia pelo limiar de similaridade

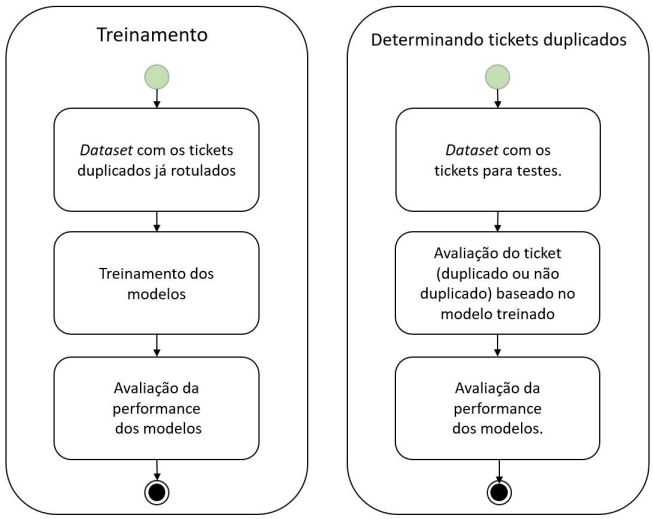



Fonte: Elaborada pelo Autor

## 5.4 Execução dos algoritmos de aprendizado de máquina

Na atividade de execução dos algoritmos de Aprendizado de Máquina, o conjunto total de dados, foi aleatoriamente dividido em dois subconjuntos, sendo que primeiro subconjunto foi utilizado para treinamento do modelo e o segundo subconjunto foi utilizado para a execução da classificação. Foram testados 4 (quatro) cenários distintos (cenários A, B, C e D), e para cada cenário, foram testados todos os algoritmos selecionados para esse experimento.

Figura 27 – Abordagem de treinamento e testes do experimento




Fonte: Elaborada pelo Autor

A Figura 27 demonstra a visão geral da abordagem utilizada no experimento. O Experimento foi dividido em 2 (duas) etapas sendo a primeira o treinamento dos modelos nos diferentes cenários (A, B, C e D) utilizando o *dataset* com os tickets duplicados já

rotulados/identificados previamente. Na segunda etapa foi realizado os testes dos modelos já treinados na etapa 1, foram executados os algoritmos no *dataset* reservado para essa etapa, foram avaliadas as métricas de cada algoritmo/modelo nos diferentes cenários de testes. Os resultados de cada um dos algoritmos e cenários de testes são demonstrados nas tabelas 11.

Tabela 11 – Resultados dos algoritmos - Cenário de testes A

| Técnica | Treino (%) | Precisão (%) | Revocação (%) | F1 (%) | Acurácia (%) |
|---|---|---|---|---|---|
| Naive Bayes | 70 % | 44,05 | 54,05 | 39,98 | 51,98 |
| SVM | 70 % | 59,60 | 60,57 | 58,45 | 62,22 |
| LSTM | 70 % | 71,45 | 71,88 | 71,66 | 73,87 |
| BERT-Base | 70 % | **74,69** | **75,89** | **75,61** | **77,71** |

Fonte: Elaborada pelo Autor

A Tabela 11 demonstra os resultados obtidos na execução dos algoritmos utilizando 70% dos registros do *dataset* para treinamento dos modelos e 30% dos registros para testes dos modelos. Esse cenário de testes foi utilizado a fim de comparar os resultados das métricas dos algoritmos com outras publicações que também utilizaram o mesmo cenário de testes [Pikies and Ali, 2019, Paramesh et al., 2018].

São demonstrados os resultados das métricas ordenando os algoritmos pela taxa de acurácia de cada modelo testado, sendo o primeiro na lista o modelo com menor acurácia, e o último e em negrito o modelo com maior acurácia.

No cenário A, BERT-Base supera LSTM em 3,84%, SVM em 15,49% e Naive Bayes em 25,73% sendo portanto, no cenário de testes A, o algoritmo que demonstrou as melhores métricas de acurácia e precisão para detecção de *tickets* duplicados.

Tabela 12 – Resultados dos algoritmos - Cenário de testes B

| Técnica | Treino (%) | Precisão (%) | Revocação (%) | F1 (%) | Acurácia (%) |
|---|---|---|---|---|---|
| Naive Bayes | 80 % | 44,47 | 54,57 | 40,32 | 52,41 |
| SVM | 80 % | 60,24 | 61,17 | 59,03 | 62,81 |
| LSTM | 80 % | 72,14 | 72,60 | 72,42 | 74,61 |
| BERT-Base | 80 % | **77,00** | **78,23** | **77,95** | **80,12** |

Fonte: Elaborada pelo Autor

No cenário B, demonstrado na tabela 12 BERT-Base também demonstrou as melhores métricas de acurácia e precisão para detecção de *tickets* duplicados, superando LSTM em 5,51%, SVM em 17,31% e Naive Bayes em 27,71%. Observa-se pelos resultados apresentados que BERT-Base apresenta um aumento nas métricas de acurácia (aumento de 2,40%) e precisão (aumento de 2,31%) em relação ao cenário A.

Tabela 13 – Resultados dos algoritmos - Cenário de testes C

| Técnica | Treino (%) | Precisão (%) | Revocação (%) | F1 (%) | Acurácia (%) |
|---|---|---|---|---|---|
| Naive Bayes | 90 % | 44,94 | 55,14 | 40,80 | 53,03 |
| SVM | 90 % | 60,81 | 61,80 | 59,64 | 63,49 |
| LSTM | 90 % | 72,90 | 73,34 | 74,20 | 75,38 |
| BERT-Base | 90 % | **79,38** | **80,65** | **80,36** | **82,59** |

Fonte: Elaborada pelo Autor

No cenário C, demonstrado na tabela 13, referente aos algoritmos selecionados BERT-Base continua demonstrando as melhores métricas de acurácia e precisão para detecção de *tickets* duplicados, superando LSTM em 7,21%, SVM em 19,11% e Naive Bayes em 29,56%. Observa-se pelos resultados apresentados que BERT-Base apresenta um aumento nas métricas de acurácia (aumento de 2,48%) e precisão (aumento de 2,38%) em relação ao cenário B.

Tabela 14 – Resultados dos algoritmos - Cenário de testes D

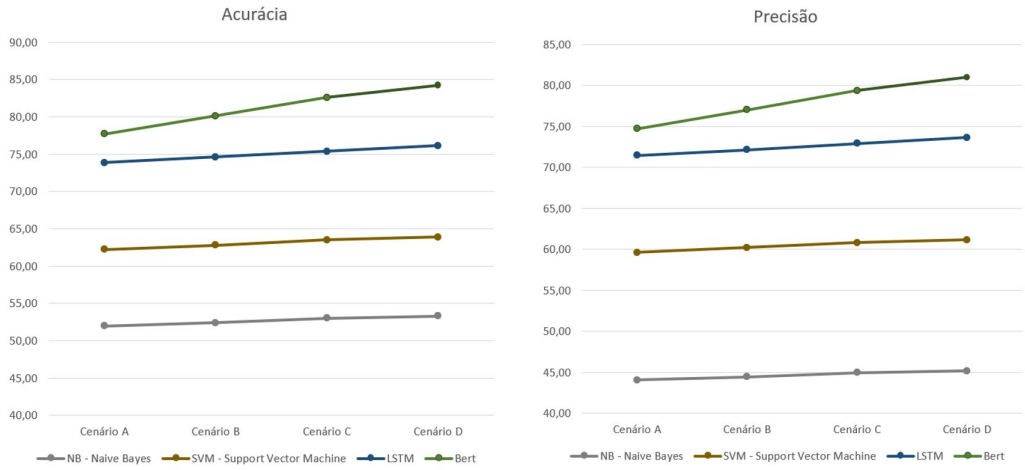
| Técnica | Treino (%) | Precisão (%) | Revocação (%) | F1 (%) | Acurácia (%) |
|---|---|---|---|---|---|
| Naive Bayes | 95 % | 45,17 | 55,42 | 41,00 | 53,80 |
| SVM | 95 % | 61,18 | 62,17 | 60,00 | 63,87 |
| LSTM | 95 % | 73,64 | 74,08 | 75,98 | 76,16 |
| BERT-Base | 95 % | **81,00** | **82,30** | **82,94** | **84,28** |

Fonte: Elaborada pelo Autor

No cenário D, demonstrado na tabela 14, BERT-Base também demonstra as melhores métricas de acurácia e precisão para detecção de *tickets* duplicados, superando LSTM em 8,13%, SVM em 20,41% e Naive Bayes em 30,98%. Também se observa, pelos resultados apresentados, que BERT-Base apresenta um aumento nas métricas de acurácia (aumento de 1,69%) e precisão (aumento de 1,62%) em relação ao cenário C.

A Figura 28 demonstra o gráfico das métricas de acurácia e precisão dos algoritmos

Figura 28 – Acurácia e precisão dos modelos nos cenários de testes



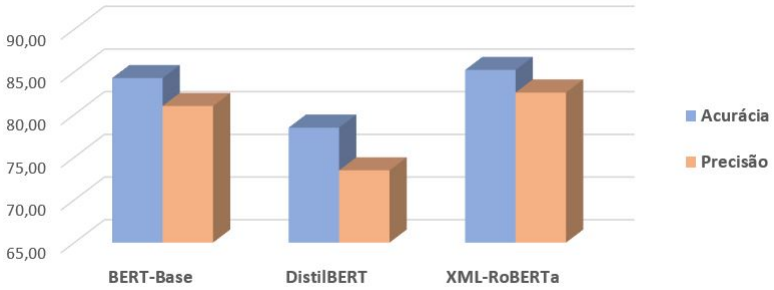

Fonte: Elaborada pelo Autor

avaliados nos diferentes cenários de testes. Observa-se que o algoritmo baseado em BERT (BERT-Base) supera Naive Bayes, SVM e LSTM em diferentes cenários e que o volume de dados de treinamento impacta o desempenho de todos os algoritmos.

A Partir dos resultados encontrados por esse experimento, foram também experimentadas outras 2 (duas) abordagens multilíngues baseadas em BERT, DistilBERT [Sanh et al., 2019] e roBERTa [Liu et al., 2019].

A Figura 29 mostra o gráfico do resultado do experimento aplicado ao cenário D (95% dos registros para testes) utilizando as técnicas baseadas em *transformer* multilíngue para Processamento de Linguagem Natural em português. Os valores das métricas de cada uma das abordagens estão demonstrados na tabela 15.

Figura 29 – Resultado do experimento para as abordagens baseadas em BERT




Fonte: Elaborada pelo Autor

Tabela 15 – Resultados das abordagens multilíngues baseadas em BERT

| Técnica | Treino (%) | Precisão (%) | Revocação (%) | F1 (%) | Acurácia (%) |
|---|---|---|---|---|---|
| DistilBERT | 95 % | 73,47 | 79,00 | 75,95 | 78,47 |
| BERT-Base | 95 % | 81,00 | 82,30 | 82,00 | 84,28 |
| XML-RoBERTa | 95 % | **82,58** | **83,47** | **81,62** | **85,23** |

Fonte: Elaborada pelo Autor

Conforme apresentado na tabela 15, observa-se que XML-RoBERTa supera BERT e DistilBERT sendo que XM-RoBERTa supera BERT-Base em 1,58 % na precisão, 1,77 % na revocação e 2,09 % no F1, também superando DistilBERT em 9,11 % em precisão, 5,07 % em revocação e 8,14 % em F1, considerando o cenário D com 95 % dos registros como treinamento dos algoritmos.

Os resultados alcançados em cada modelo preditivo gerado, variaram de acordo com a técnica e hiperparâmetros utilizados. Para maximizar os resultados, foram utilizadas técnicas de hiperparametrização através de grid search. A Tabela 16 mostra os principais parâmetros utilizados em cada modelo do experimento.

Tabela 16 – Parâmetros do experimento

| Algoritmo | Parâmetros |
|---|---|
| Naive Bayes | alpha=1.0, fit_prior=True, class_prior=None |
| SVM | C=1.0, kernel='linear', degree=3, gamma='auto' |
| LSTM | vocab_size=1000,embedding_dim=64,max_length=None |
| BERT-Base | epochs=3, batch_size=32, eval_batch_size=32 |

Fonte: Elaborada pelo Autor

As abordagens baseadas em *transformers*, particularmente BERT, geralmente requerem poucas interações para convergir em um modelo capaz de fornecer resultados eficientes [Wolf et al., 2020]. Foram testadas diferentes épocas para definir este parâmetro como 3, para um melhor equilíbrio entre o tempo de treinamento e o desempenho do modelo.

Os resultados do BERT mostram que é um modelo de linguagem treinado bidirecionalmente que possui um entendimento mais profundo do contexto e fluxo de linguagem do que modelos de linguagem de direção única [Devlin et al., 2018]. Dessa forma, BERT aprende as relações contextuais entre as palavras no texto do *ticket*.

Baseado em publicações anteriores, os modelos baseados em BERT apresentam as melhores métricas para a atividade de Processamento de Linguagem Natural. visto que os modelos são organizados em 2 (duas) etapas, na primeira etapa o modelo é treinado exaustivamente para que seja capaz de compreender uma língua, no que é chamado na área de *modelagem da linguagem*, nessa etapa o modelo aprende a inferir uma palavra através do contexto da sentença onde está inserida, assim como entender as relações entre duas sentenças sequenciais. Na segunda etapa depois que já foi treinado para entender a língua ele foi utilizado para a classificação de textos. Esse é o processo de *transfer learning*, e trata da transferência do aprendizado genérico feito na primeira etapa para a tarefa específica de detecção de *tickets* duplicados. Para isso, o modelo usa a parte do *encoder*, com seus parâmetros "calibrados" na etapa 1, e depois, passa essa representação do texto para um classificador, o qual foi treinado na tarefa do experimento [Devlin et al., 2018, Xiao et al., 2019].

Cada modelo pré-treinados de BERT (BERT-base [Devlin et al., 2018], DistilBERT [Sanh et al., 2019] e roBERTa [Liu et al., 2019] utilizado por esse experimento, estão disponíveis gratuitamente na internet sendo que os detalhes de cada modelo estão demonstrados na tabela 17.

Tabela 17 – Modelos BERT pré-treinados

| Modelo | Detalhes do modelo |
|---|---|
| BERT-base | 12-layer, 768-hidden, 12-heads, 110M parameters. Treinado em 104 principais idiomas com os artigos das maiores Wikipedias |
| DistilBERT | 6-layer, 768-hidden, 12-heads, 66M parameters DistilBERT utiliza um modelo "destilado"a arquitetura base do BERT-base |
| RoBERTa | 12-layer, 768-hidden, 12-heads, 125M parameters RoBERTa utiliza a arquitetura base do BERT-base |

Fonte: Elaborada pelo Autor

# 6 Conclusões

Esta dissertação teve como objetivo principal avaliar os resultados das técnicas de Aprendizado de Máquina utilizados na literatura para classificação e recuperação de *tickets* em bases de dados de língua estrangeira quando utilizados em base de dados de língua portuguesa (Brasil).

Para se atingir o objetivo principal, foram-se definidos 3 (três) objetivos específicos, que foram executados no processo de desenvolvimento dessa pesquisa.

O Primeiro objetivo específico buscou identificar as técnicas de Aprendizado de Máquina utilizadas na literatura para classificação e recuperação de *ticket* em língua estrangeira. Esse objetivo foi alcançado através de da revisão da literatura identificando os trabalhos científicos que desenvolveram experimentos com essa finalidade, sendo identificadas as 4 (quatro) técnicas que apresentaram as melhores métricas de desempenho (Naive Bayes, SVM, LSTM e BERT-Base). Os trabalhos relacionados estão descritos no capítulo 3.

Para se atingir o segundo objetivo específico desse trabalho, foi desenvolvido um experimento que aplicou as técnicas de Aprendizado de Máquina identificadas na literatura em uma base de dados de *ticket* de uma empresa brasileira de prestação de serviços em TI, na língua portuguesa. Esse experimento está detalhado no capítulo 5.

Por fim, para se atingir o terceiro e último objetivo específico, as técnicas de Aprendizado de Máquina foram classificadas pelas suas métricas de acurácia e precisão na atividade de detecção e recuperação de *tickets* duplicados. Os resultados do experimento demonstram as métricas de cada técnica utilizada na detecção e recuperação de tickets duplicados e estão detalhados na seção 5.4.

Referente aos modelos de base, o modelo baseado em Naive Bayes apresentou 53,30% de acurácia e 45,17% de precisão, este desempenho está próximo ao desempenho relatado na literatura para modelos de classificação ingênuos. Sendo que esse modelo fornece uma linha de base pela qual todos os outros modelos podem ser considerados habilidosos ou não [Tang et al., 2016]. Em relação ao modelo implementado com SVM, o são alcançados 63,87% de acurácia e 61,18% de precisão, este desempenho também está coerente ao desempenho de experimentos anteriores relatados na literatura [Shafiabady et al., 2016].

O Modelo LSTM apresenta desempenho superior aos dois modelos de base, 80,15% de acurácia e 73,64% de precisão, sendo justificado pois esse modelo tem em sua arquitetura redes de memória de longo prazo que são um tipo especial de Redes Neurais

Recorrentes. Esses modelos de redes neurais profundas são utilizadas na literatura para processamento de linguagem natural, sendo que trabalhos anteriores referente à classificação de *tickets*, o desempenho apresentado se mostra equivalente [Greff et al., 2017].

O Melhor desempenho apresentado nos modelos com a implementação das técnicas avaliadas por este trabalho são as abordagens baseadas em BERT (DistilBERT 78,47% de acurácia e 73,47% de precisão, BERT-Base 84,28% de acurácia e 81,00 de precisão e XML-RoBERTa 85,23% de acurácia e 82,58% de precisão.

O resultado deste trabalho fortalece a tese de que o modelo BERT é indicado como o mais eficaz, uma vez que esse modelo é classificado pela literatura como estado da arte. Seu desempenho superior é justificado pelo seu conhecimento prévio das estruturas da linguagem, visto que seu treinamento utiliza aprendizado semi-supervisionado em grandes bases de dados [Devlin et al., 2018, Xiao et al., 2019].

Acredita-se que o *dataset* utilizado por esse trabalho, favoreceu os resultados encontrados, uma vez que os dados já encontravam todos estruturados, os campos de resumo e descrição dos *tickets* foram redigidos por usuários utilizadores de soluções de tecnologia (hardware e software) não sendo identificados números relevantes de erros de digitação. Somando-se a esses fatores, os assuntos tratados nos *tickets* se referiam exclusivamente à questões relacionadas ao domínio de soluções de tecnologia ofertadas pela empresa de prestação de serviços que disponibilizou o *dataset*.

Os resultados apresentados nesta pesquisa corroboram os dados apresentados por pesquisas semelhantes realizadas em repositório de tickets de língua inglesa sendo que os resultados apresentados na execução do experimento vão de acordo com o esperado, ressaltando o destaque de desempenho para as abordagens baseadas em BERT acredita-se ser viável a implementação desses modelos em ferramentas automatizadas de detecção e recuperação de *tickets* duplicados, visando sobretudo a redução do esforço manual da equipe de primeiro nível da central de serviços e melhoria dos indicadores de desempenho e satisfação dos usuários dos serviços.

Referente ao *dataset* avaliado por esse trabalho, nota-se que para os *tickets* pertencentes à categoria *Pergunta/Dúvida* é possível implementar um modelo automatizado de detecção e recuperação de *tickets* duplicados e sugerir automaticamente uma provável solução para a ocorrência baseado no histórico de *tickets* duplicados recuperados na base de dados.

## 6.1 Limitações da pesquisa

Acredita-se que serão necessários mais testes antes que estes modelos possam ser utilizados em aplicações do mundo real. Para isso, sugere-se aplicar estas mesmas técnicas em outros conjuntos de repositórios de *tickets* na língua portuguesa, se possível com um maior número de amostras e em organizações com distintos tipos de serviços para reavaliação dos resultados e possivelmente ajustes finos nos parâmetros dos modelos.

Duas das técnicas utilizadas neste trabalho (Naive Bayes e SVM) são consideradas linha de base *baseline*, contudo são técnicas já experimentadas na literatura para o tipo de problema de classificação de *tickets*. Foi também implementada a técnica BERT considerada estado da arte para Processamento de Linguagem Natural, entretanto há na literatura outras técnicas já experimentadas para classificação de textos utilizando *Redes Neurais Artificiais*, *Redes Neurais Recorrentes*, *Redes Neurais Convolutivas* [Goodfellow et al., 2016, Yalur, 2019] que poderão também ser implementadas no contexto de *tickets*.

## 6.2 Trabalhos futuros

Este trabalho se concentrou exclusivamente na detecção e recuperação de *tickets* duplicados em uma base de dados de uma empresa de prestação de serviços de TI, usando técnicas de Aprendizado de Máquina, contudo, há na literatura outras técnicas de *Machine Learning* e técnicas de *Deep Learning* utilizadas na classificação de texto e que poderão ser exploradas no contexto de *tickets* e poderão contribuir para a melhoria da qualidade preditiva dos modelos classificadores.

Podem ser exploradas maneiras de portar modelos estatísticos existentes induzidos a partir do *dataset* de uma empresa específica para outras aplicações alavancando o corpo de trabalho existente sobre transferência de aprendizagem entre domínios.

Sugere-se também como trabalhos futuros, a detecção e recuperação de *tickets* semelhantes, que não são considerados réplicas exatas por apresentarem menor similaridade. A identificação dessas ocorrências poderá auxiliar os atendentes da central de serviços na identificação de uma solução já adotada anteriormente e que tenha o contexto próximo da ocorrência do *ticket* que está sendo analisado.

# Referências

[Aho et al., 2007] Aho, A. V., Sethi, R., and Ullman, J. D. (2007). *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Longman Publishing Co., Inc., USA.

[Aranha et al., 2007] Aranha, C. N., Vellasco, M., and Passos, E. (2007). Uma abordagem de pré-processamento automático para mineração de textos em português: Sob o enfoque da inteligência computacional. pages 167–175.

[Beam and Kohane, 2018] Beam, A. L. and Kohane, I. S. (2018). Big Data and Machine Learning in Health Care. *JAMA*, 319(13):1317–1318.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). *A neural probabilistic language model*. Journal of Machine Learning Research.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, NY. Softcover published in 2016.

[Bon, 2002] Bon, J. v. (2002). *IT Service Management: An Introduction*. ServiceNow® ITSM.

[Budhiraja et al., 2018] Budhiraja, A., Dutta, K., Shrivastava, M., and Reddy, R. (2018). Towards word embeddings for improved duplicate bug report retrieval in software repositories. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '18, page 167–170, New York, NY, USA. Association for Computing Machinery.

[Buitinck et al., 2013] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). Api design for machine learning software: experiences from the scikit-learn project. *ArXiv*, abs/1309.0238.

[Bulbul and Unsal, 2011] Bulbul, H. I. and Unsal, O. (2011). *Comparison of Classification Techniques used in Machine Learning as Applied on Vocational Guidance Data*, volume 2. IEEE Press.

[C. et al., 2016] C., O., Nepomuceno, E., and Oliveira, M. (2016). Análise de sensibilidade dos parâmetros do aprendizado por reforço na solução do problema do caixeiro viajante: Modelagem via superfície de resposta. *Conference: Anais do CBA 2016 - XXI Congresso Brasileiro de Automática*.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(75):2493–2537.

[Covões and Hruschka, 2011] Covões, T. F. and Hruschka, E. R. (2011). Towards improving cluster-based feature selection with a simplified silhouette filter. *Information Sciences*, 181(18):3766–3782.

[Córdova and Silveira, 2009] Córdova, F. P. and Silveira, D. T. (2009). *A pesquisa científica.* UFRGS Editora.

[Deshmukh et al., 2017] Deshmukh, J., Annervaz, K. M., Podder, S., Sengupta, S., and Dubash, N. (2017). Towards accurate duplicate bug retrieval using deep learning techniques. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 115–124.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. pages 112–127.

[Dias and Malheiros, 2005] Dias, M. and Malheiros, M. (2005). Extração automática de palavras-chave de textos da língua portuguesa. *Universidade do Vale do Taquari - UNIVATES*, pages 37–45.

[dos Santos and Gatti, 2014] dos Santos, C. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

[Dumais, 2004] Dumais, S. T. (2004). Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230.

[Eckstein et al., 2016] Eckstein, L., Kuehl, N., and Satzger, G. (2016). Towards extracting customer needs from incident tickets in it services. In *2016 IEEE 18th Conference on Business Informatics (CBI)*, volume 01, pages 200–207.

[Foody, 2002] Foody, G. M. (2002). Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80(1):185–201.

[Geron, 2017] Geron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* OReilly Media, Sebastopol, CA, USA.

[Gil, 2007] Gil, A. C. (2007). *Como elaborar projetos de pesquisa.* Editora Atlas S.A.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. pages 1–9. http://www.deeplearningbook.org.

[Greff et al., 2017] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

[Gunasekara and Haddela, 2018] Gunasekara, S. V. S. and Haddela, P. S. (2018). Context aware stopwords for sinhala text classification. In *2018 National Information Technology Conference (NITC)*, pages 1–6.

[Gunawan, 2019] Gunawan, H. (2019). Strategic management for it services using the information technology infrastructure library (itil) framework. In *2019 International Conference on Information Management and Technology (ICIMTech)*, volume 1, pages 362–366.

[Guoxiang and Linlin, 2011] Guoxiang, D. and Linlin, J. (2011). The lexical approach for language teaching based on the corpus language analysis. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 665–668.

[Guyon et al., 2002] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1–3):389–422.

[Hall, 2000] Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, ICML '00, page 359–366, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Haykin, 1994] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, USA, 1st edition.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *ArXiv*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

[Jan et al., 2013] Jan, E., Ni, J., Ge, N., Ayachitula, N., and Zhang, X. (2013). A statistical machine learning approach for ticket mining in it service delivery. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 541–546.

[Kadar et al., 2011] Kadar, C., Wiesmann, D., Iria, J., Husemann, D., and Lucic, M. (2011). Automatic classification of change requests for improved it service quality. In *2011 Annual SRII Global Conference*, pages 430–439.

[Kalchbrenner et al., 2014] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 1:655–665.

[Kallis et al., 2019] Kallis, R., Di Sorbo, A., Canfora, G., and Panichella, S. (2019). Ticket tagger: Machine learning driven issue classification. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 406–409.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification.

[Kurz et al., 2020] Kurz, N., Hamann, F., and Ulges, A. (2020). Neural entity linking on technical service tickets. In *2020 7th Swiss Conference on Data Science (SDS)*, pages 35–40.

[Lanyo and Wausi, 2018] Lanyo, K. and Wausi, A. (2018). A comparative study of supervised and unsupervised classifiers utilizing extractive text summarization techniques to support automated customer query question-answering. In *2018 5th International Conference on Soft Computing Machine Intelligence (ISCMI)*, pages 88–92.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2015] Lee, C., Hu, D., Feng, Z., and Yang, C. (2015). Mining temporal information to improve duplication detection on bug reports. In *2015 IIAI 4th International Congress on Advanced Applied Informatics*, pages 551–555.

[Li and Li, 2019] Li, L.-F. and Li, W. (2019). Naive bayesian automatic classification of railway service complaint text based on eigenvalue extraction. In *Tehnicki vjesnik*.

[Liu and Setiono, 1996] Liu, H. and Setiono, R. (1996). A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 30th International Conference on*

*International Conference on Machine Learning*, ICML'96, page 319–327, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

[Lyubinets et al., 2018] Lyubinets, V., Boiko, T., and Nicholas, D. (2018). Automated labeling of bugs and tickets using attention-based mechanisms in recurrent neural networks. In *2018 IEEE 2nd International Conference on Data Stream Mining Processing (DSMP)*, pages 271–275.

[Marquis, 2010] Marquis, H. (2010). *How to Classify Incidents*. ITSM Solution.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity*, page 115–133. MIT Press, Cambridge, MA, USA.

[Mikolov et al., 2013] Mikolov, T., Corrado, G., Chen, K., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on International Conference on Machine Learning*, pages 1–12.

[Montgomery and Damian, 2017] Montgomery, L. and Damian, D. (2017). What do support analysts know about their customers? on the study and prediction of support ticket escalations in large software organizations. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 362–371.

[Mubeen et al., 2011] Mubeen, S., Qaseem, M. S., and Govardhan, A. (2011). Usage of distinctive classifiers for text categorization using distributional features. In *2011 Annual IEEE India Conference*, pages 1–5.

[Orengo and Huyck, 2001] Orengo, V. M. and Huyck, C. (2001). A stemming algorithm for the portuguese language. In *Proceedings Eighth Symposium on String Processing and Information Retrieval*, pages 186–193.

[Paramesh et al., 2018] Paramesh, S. P., Ramya, C., and Shreedhara, K. S. (2018). Classifying the unstructured it service desk tickets using ensemble of classifiers. In *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, pages 221–227.

[Parmar et al., 2018] Parmar, P. S., Biju, P. K., Shankar, M., and Kadiresan, N. (2018). Multiclass text classification and analytics for improving customer support response through different classifiers. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 538–542.

[Pearson, 1901] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

[Phetrungnapha and Senivongse, 2019] Phetrungnapha, K. and Senivongse, T. (2019). Classification of mobile application user reviews for generating tickets on issue tracking system. In *2019 12th International Conference on Information Communication Technology and System (ICTS)*, pages 229–234.

[Pikies and Ali, 2019] Pikies, M. and Ali, J. (2019). String similarity algorithms for a ticket classification system. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 36–41.

[Pushpa et al., 2017] Pushpa, S. K., Manjunath, T. N., Mrunal, T. V., Singh, A., and Suhas, C. (2017). Class result prediction using machine learning. In *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pages 1208–1212.

[Qamili et al., 2018] Qamili, R., Shabani, S., and Schneider, J. (2018). An intelligent framework for issue ticketing system based on machine learning. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 79–86.

[Reshma and Remya, 2017] Reshma, E. U. and Remya, P. C. (2017). A review of different approaches in natural language interfaces to databases. In *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, pages 801–804.

[Rodrigues, 2012] Rodrigues, A. (2012). *How to Classify Incidents*. IEEE Press.

[Roy et al., 2016] Roy, S., Yan, J.-J., Budhiraja, N., and Lim, A. (2016). Recovering resolutions for application maintenance incidents. In *Conference: 2016 IEEE International Conference on Services Computing (SCC)*, pages 617–624.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

[Sasso et al., 2016] Sasso, T. D., Mocci, A., and Lanza, M. (2016). What makes a satisficing bug report? In *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 164–174.

[Shafiabady et al., 2016] Shafiabady, N., Lee, L., Rajkumar, R., Kallimani, V., Akram, N. A., and Isa, D. (2016). Using unsupervised clustering approach to train the support

vector machine for text classification. *Neurocomputing*, 211:4 – 10. SI: Recent Advances in SVM.

[Shah and Patel, 2016] Shah, F. P. and Patel, V. (2016). A review on feature selection and feature extraction for text classification. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2264–2268.

[Shen et al., 2014] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, page 373–374, New York, NY, USA. Association for Computing Machinery.

[Son et al., 2014] Son, G., Hazlewood, V., and Peterson, G. D. (2014). On automating xsede user ticket classification. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, XSEDE '14, New York, NY, USA. Association for Computing Machinery.

[Suhairi and Gaol, 2013] Suhairi, K. and Gaol, F. (2013). The measurement of optimization performance of managed service division with itil framework using statistical process control. *J. Networks*, 8:518–529.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

[Tang et al., 2016] Tang, B., Kay, S., and He, H. (2016). Toward optimal feature selection in naive bayes for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2508–2521.

[Van Rijsbergen et al., 1980] Van Rijsbergen, C. J., Robertson, S. E. (Stephen Edward), ., and Porter, M. F. M. F. (1980). New models in probabilistic information retrieval. *The London, British Library Research and Development Dept.* Includes bibliographical references.

[Venkataraman, 2016] Venkataraman, A. (2016). Building accurate text classifiers. *International Journal of Digital Information and Wireless Communications*, 6:1–47. Report.

[Venkatesh and Ranjitha, 2018] Venkatesh and Ranjitha, K. V. (2018). Classification and optimization scheme for text data using machine learning naïve bayes classifier. In *2018 IEEE World Symposium on Communication Engineering (WSCE)*, pages 33–36.

[Wang et al., 2015] Wang, Y., Fu, W., Sui, A., and Ding, Y. (2015). Comparison of four text classifiers on movie reviews. In *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*, pages 495–498.

[Wolf et al., 2020] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

[Xiao et al., 2019] Xiao, f. M., Zhiguo, W., Patrick, N., Ramesh, N., and Bing, X. (2019). Universal text representation from bert: An empirical study. *ArXiv*, abs/1910.07973.

[Yalur, 2019] Yalur, T. (2019). Interperforming in artificial intelligence: question of 'natural' in machine learning and recurrent neural networks. *AI and SOCIETY*, pages 237–246.

[Yandri et al., 2019] Yandri, R., Suharjito, S., Utama, D., and Zahra, A. (2019). Evaluation model for the implementation of information technology service management using fuzzy itil. *Procedia Computer Science*, 157:290–297.

[Yang et al., 2016] Yang, X., Lo, D., Xia, X., Bao, L., and Sun, J. (2016). Combining word embedding with information retrieval to recommend similar bug reports. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 127–137.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Yih et al., 2011] Yih, W.-t., Toutanova, K., Platt, J. C., and Meek, C. (2011). Learning discriminative projections for text similarity measures. In *Proceedings of the 15th Conference on Computational Natural Language Learning*, pages 247–256, Portland, Oregon, USA. Association for Computational Linguistics.

[Zhang et al., 2019] Zhang, D., Yang, Z., and Liu, X. (2019). A lexical conversion as preprocessing method for text intention classification. In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pages 493–496.

[Zighed et al., 2000] Zighed, D. A., Komorowski, J., and Zytkow, J. (2000). *Principles of Data Mining and Knowledge Discovery*. Springer-Verlag, Berlin, Heidelberg.