ENCAMINHAMENTO DE *TICKETS* QUE EXIGEM FORMAÇÃO DE GRUPOS POR CLASSIFICAÇÃO DE TEXTO

Jones Marques Augusto

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2019

ENCAMINHAMENTO DE *TICKETS* QUE EXIGEM FORMAÇÃO DE GRUPOS POR CLASSIFICAÇÃO DE TEXTO

Jones Marques Augusto

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Alexandre de Assis Bento Lima , D.Sc.

Prof. Leandro Guimarães Marques Alvim, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2019

Augusto, Jones Marques

Encaminhamento de *Tickets* que Exigem Formação de Grupos por Classificação de Texto/Jones Marques Augusto. – Rio de Janeiro: UFRJ/COPPE, 2019.

XI, 50 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 44 – 47.

1. Text Classification. 2. Ticket Routing. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico a toda minha família que sempre me apoiou.

Agradecimentos

Agradeço à minha família, que sempre me apoiou e orientou a buscar algo melhor. Preciso dar um destaque especial a minha esposa Elaine Tady, por estar comigo sempre me ajudando a resolver todos os problemas existentes. Para mim foi ótimo saber que sempre tive alguém com quem contar.

Agradeço é claro também a todos os amigos que conheci nessa caminhada, grupos de estudos, grupos de trabalhos e etc. Sem eles o caminho seria muito mais difícil, muitas histórias e risadas que tornaram essa vida de estudante muito mais fácil.

Agradeço ao meu orientador/professor Xexéo, em primeiro lugar por ter me aceitado como seu orientando e em segundo lugar por ter dado todo suporte que necessitei ao produzir esse trabalho. Também tenho que destacar os professores do curso de Mestrado do PESC/UFRJ.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ENCAMINHAMENTO DE *TICKETS* QUE EXIGEM FORMAÇÃO DE GRUPOS POR CLASSIFICAÇÃO DE TEXTO

Jones Marques Augusto

Setembro/2019

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Com o crescimento da Tecnologia de Informação (TI) é cada vez mais importante para a economia mundial o uso de serviços de TI, a comercialização desses serviços tornou-se estratégica para diversas empresas e organizações. Esses provedores têm o desafio de manter a infraestrutura tecnológica em bom estado de funcionamento, pois precisam garantir que seus serviços estarão sempre disponíveis a fim de manter seus clientes satisfeitos. Com o acesso facilitado graças a *Internet*, os usuários normalmente têm diversas opções para realizar a tarefa desejada e podem migrar caso o serviço não os atendam.

Uma vez que o mundo se tornou dependente de sistemas de TI, a queda de qualidade ou indisponibilidade destes serviços não são boas para os negócios. Com isso, a solução dos incidentes que causam a maioria desses problemas é um dos pontos importantes da gestão de serviços de TI. Buscando reduzir a carga de trabalho envolvida no gerenciamento de incidentes, um sistema que encaminha incidentes registrados para as áreas que têm a capacidade de resolvê-los é criado e testado com incidentes reais coletados de uma empresa brasileira provedora de serviços de TI.

Para isto, técnicas de classificação de texto são analisadas sobre duas abordagens diferentes. A primeira encaminha os incidentes que são resolvidos por apenas um grupo de especialistas e a segunda abordagem trabalha com os incidentes que necessitam de mais de um grupo de especialistas para serem resolvidos. Os experimentos propostos mostram bons resultados na redução da carga de trabalho e acurácia na classificação.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TICKET ROUTING THAT REQUIRES GROUPS BY TEXT CLASSIFICATION

Jones Marques Augusto

September/2019

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

With the growth of Information Technology (IT) the use of IT services is becoming more important for the world economy, the commercialization of these services has become strategic for many companies and organizations. These providers are challenged to keep the technology infractutures up and running as they need to ensure that their services are always available to keep their customers happy. With the easy access thanks to the Internet, users usually have several options to perform the desired task and can migrate if the service does not meet them.

Since the world has become dependent on IT systems, the poor quality or unavailability of these services is not good for business. Thus, resolving the incidents that cause most of these problems is one of the important points of IT service management. Seeking to reduce the workload involved in incident management, a system that routes recorded incidents to areas that have the ability to address them is created and tested using actual incidents collected from a Brazilian IT service provider.

To do so, text classification techniques are employed using two different approaches. The first addresses incidents that are solved by only one expert group, and the second approach works with incidents that need more than one expert group to solve. The proposed experiments show good results in workload reduction and classification accuracy.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

A importância da Tecnologia da Informação (TI) cresce à medida que a tecnologia avança e chega em todas as camadas da sociedade (Marrone et al., 2014). Nos negócios, as empresas privadas dos Estados Unidos gastam mais da metade de seu dinheiro investindo em TI (Marrone et al., 2014).

Com a ampliação do domínio da TI, o mundo entrou em um período que os serviços se tornaram a principal força da economia, a maior fonte de empregos e constituindo a maior parcela do PIB na maioria dos países ao redor do mundo (Dahlgaard-Park, 2015). Com o crescimento da TI e a ascensão da comercialização de serviços, aparecem os serviços de TI. Existindo em uma interseção entre as duas áreas, os serviços de TI são bastante valiosos, já que as ferramentas usadas para acessá-los estão cada vez mais acessíveis a qualquer pessoa (computadores, celulares e tablets) (Dahlgaard-Park, 2015).

Varejistas, como *Amazon*[1] e *Walmart*[2], executam diversas transações de vendas *on-line* e oito em cada dez americanos são compradores *on-line* (Smith and Anderson, 2016). Empresas como *Google*[3] e *Microsoft*[4] oferecem aos seus usuários a oportunidade de armazenar arquivos pessoais (fotos, músicas, documentos, etc.) em servidores localizados dentro de seus *data centers*. O armazenamento em nuvem é tão comumente utilizado que em 2016 ultrapassou o compartilhamento de arquivos como a maior fonte de tráfego *upstream* (isto é, dados que fluem de computadores para a rede) durante o período de pico de acesso norte-americano (Sandvine, 2016).

[1] http://www.amazon.com/
[2] http://www.walmart.com/
[3] http://www.google.com/
[4] http://www.microsoft.com/

Serviços de *streaming* como *Netflix*[5] e *Youtube*[6] possuem vastos catálogos de vídeos que podem ser acessados por usuários em diferentes plataformas, e o *streaming* de áudio e vídeo representa 71% do tráfego de rede noturno na América do Norte (Sandvine, 2016). Empresas de várias áreas empregam ferramentas e aplicativos para produzir e agregar valor a seus clientes, assim como os governos ao redor do mundo que disponibilizam serviços essenciais para seus cidadãos na *Internet* (Zhou, 2015).

Os sistemas que permitem os usuários executarem essas atividades dependem de uma boa infraestrutura física (como servidores, roteadores e etc). A falta de manutenção dessa infraestrutura pode causar problemas indesejáveis, que podem variar de erros pontuais e lentidão do serviço até a indisponibilidade completa (Zhou, 2015). Em um ambiente econômico de alta competitividade, graças à *Internet*, em que os usuários podem escolher livremente a partir de um amplo conjunto de opções, é essencial manter o baixo custo e suporte constantes dado a complexidade dos ambientes tecnológicos que existem por trás da maioria dos serviços de TI (Zhou, 2015).

Os problemas relacionados à infraestrutura podem fazer com que clientes insatisfeitos e frustrados deixem de pagar ou consumir o que é fornecido por uma empresa e migrem para rivais (Zhou, 2015). Portanto, é fundamental encontrar maneiras de reduzir as taxas com que esses problemas aparecem e agir rapidamente sobre eles quando ocorrem, de modo que seus efeitos não sejam percebidos. O tempo de inatividade do serviço é percebido sempre de forma negativa: afeta os negócios, faz com que os usuários fiquem infelizes e pode diminuir a qualidade da imagem de uma empresa quando comparada com a de seus concorrentes (Zhou, 2015).

Com o objetivo de estabelecer uma série de práticas para garantir que os serviços de TI agreguem valor ao negócio, foi criada a *ITIL (Information Technology Infrastructure Library)*. Dentro dessa organização, os incidentes são tratados pelo processo de Gerenciamento de Incidentes e pela função *Service Desk* (ITIL, 2007).

O primeiro define uma série de etapas para detectar, registrar, classificar e gerenciar o incidente em todo o seu ciclo de vida (ITIL, 2007). Enquanto isso, o último serve como ponto de entrada para todos os incidentes (ITIL, 2007). Os membros da central de atendimento são contatados por usuários que estão enfrentando problemas e é responsabilidade do *Service Desk* criar um *ticket* de incidente registrando o problema em um sistema, descrevendo os sintomas relatados e tentar resolver o *ticket* ou optar por direcioná-lo para outras áreas quando o incidente se mostrar muito complexo, exigindo tratamento por uma equipe mais especializada (Moharreri et al., 2016).

[5] `http://www.netflix.com/`
[6] `http://www.youtube.com/`

A capacidade do *Service Desk* de resolver os *tickets* em tempo hábil determina, em grande parte, sua vantagem competitiva (Miao et al., 2010). Para gerenciar a resolução de *tickets* com eficácia, os especialistas humanos costumam ser organizados em grupos, cada um com experiência para resolver certos tipos de problemas. À medida que os sistemas de TI se tornam mais complexos, os tipos de problemas relatados se tornam mais diversificados. Encontrar um grupo de especialistas para resolver o problema descrito em um *ticket* é um desafio de longa data para os provedores de serviços de TI (Miao et al., 2010).

Um grande *Service Desk* precisa lidar, diariamente, com um grande número de *tickets* de incidentes relatando diversos problemas diferentes (Miao et al., 2010). Quando a ocorrência do usuário não tem solução já conhecida, o *Service Desk* encaminha o *ticket* para um departamento específico para que seja analisado mais profundamente e proposta uma nova solução. Muitos desses *tickets* podem ser encaminhados entre vários grupos de especialistas antes de serem transferidos para o grupo que de fato o finalizará, o que só torna mais importante esse gerenciamento de incidentes (Miao et al., 2010).

A resolução de incidentes ocorridos em sistemas de TI é essencial para empresas e ainda um desafio a ser alcançado (Moharreri et al., 2016). Muitas pesquisas aplicam modelos probabilísticos na tentativa de reduzir o número de transferências entre grupos de especialistas (Moharreri et al., 2016).

Esse trabalho apresenta um estudo de caso real de um provedor de serviços de TI brasileiro que implementa o *ITIL*. O grupo de *tickets* de incidentes que foi obtido é utilizado na análise do problema de classificação de incidentes através de duas perspectivas diferentes, a primeira classifica os *tickets* que são resolvidos por apenas um grupo de especialistas e a segunda considera todos os *tickets* que continham transições entre os diferentes grupos.

1.2 Objetivos

O objetivo desse trabalho é propor soluções (classificadores e modelos) para auxiliar nos processos e ganhar velocidade na solução dos *tickets*. Utilizando uma base de incidentes reais catalogados de uma prestadora de serviços brasileira, é feita uma análise dos processos e dos *logs* de incidentes da empresa aplicando técnicas para analisar os textos contidos nas descrições dos *tickets* e os encaminhar automaticamente para o grupo de especialistas melhor indicado para encontrar a solução do problema.

Diferentes algoritmos de classificação e modelos de análise de texto serão avaliados em busca das melhores soluções para auxiliar o *Service Desk*.

1.3 Organização

Este trabalho está divido em 6 capítulos. O capítulo 2 apresenta uma explicação sobre o *ITIL*, sua definição, regras e boas práticas. Apresenta também uma explanação sobre modelagem de processos, como identificar, categorizar e criar uma visualização clara e objetiva dos processos de uma organização.

O capítulo 3 apresenta um levantamento das técnicas utilizadas nos experimentos desse trabalho, mostrando suas principais aplicações e porque foram escolhidas para solucionar o problema apresentado. Primeiro são apresentados os tratamentos responsáveis por transformar texto livre em uma representação compreendida pelos algoritmos, em seguida são apresentados os algoritmos de classificação utilizados sobre essas representações dos textos. Por fim são apresentados tratamentos e pré-processamentos para melhorar a aplicação das técnicas.

O capítulo 4 apresenta a proposta de solução para o problema. Mostra o embasamento teórico das soluções, trabalhos relacionados, a metodologia utilizada e a forma de avaliação.

O capítulo 5 apresenta os experimentos realizados para avaliar as propostas de solução. O primeiro grupo de experimentos são aplicados sobre os *tickets* que só passaram por um único grupo de especialistas para serem resolvidos, em seguida são apresentados os experimentos que tratam os *tickets* com passagens por mais de um grupo de especialistas.

Já o capítulo 6 apresenta as conclusões finais do trabalho. Pontos positivos e negativos dos experimentos e possíveis trabalhos futuros.

Capítulo 2

Revisão

2.1 Biblioteca de Infraestrutura de Tecnologia da Informação (*ITIL*)

O *ITIL* estabelece uma estrutura para gerenciamento de serviços de uma empresa de TI. Ele nasce de uma iniciativa do governo do Reino Unido, que pretendia unificar as práticas empregadas pelas organizações que mantinham os gerenciamentos de seus serviços de TI (ITIL, 2007).

A biblioteca contém um mapeamento de todo o ciclo de vida de um produto. É dividido em cinco partes distintas que, se implantadas corretamente, ajudarão as empresas de TI a atingir suas metas e fornecer serviços de qualidade aos clientes, fornece também o conhecimento para evitar ou reduzir crises. As partes que compõem o *ITIL* espelham as diferentes fases de um serviço (ITIL, 2007).

O *ITIL* pode ser organizado em 5 etapas dentro do ciclo de vida dos serviços de uma organização (ITIL, 2007):

- Estratégia de serviço (*Service Strategy*) - Lida com os requisitos, metas e expectativas, preparando as empresas para os riscos e ações operacionais que estão envolvidos;

- Projeto de serviço (*Service Design*) - Define processos para documentar planos, treinamento, criar serviços confiáveis e que tenham a disponibilidade e o desempenho que os clientes esperam deles;

- Transição de serviço (*Service Transition*) - Assegura que a transição dos serviços que saem da etapa de design sejam realizados de maneira organizada, isso inclui a validação e o teste dos serviços, a manutenção dos registro de todos os itens de infraestrutura e suas configurações, além do planejamento de como a transição dos serviços será realizada;

- Operação do serviço (*Service Operation*) - Assegura que os níveis acordados serão cumpridos (*SLA*);

- Melhoria contínua do serviço (*Continual Service Improvement*) - Envolve todas as fases anteriores, visto que a melhoria contínua do serviço é aplicada para avaliar e refinar todas as etapas do ciclo de vida *ITIL*.

O ciclo de vida *ITIL* ainda é dividido em partes menores para detalhar melhor todos os serviços da organização (ITIL, 2007), dessas partes se destacam:

- Processo - recebe entradas, altera ativos por meio de uma série de atividades para produzir um resultado que é entregue ao cliente.

- Função - uma ou mais unidades que executam um trabalho específico. As funções assumem os papéis que lidam com os processos, bem como as atividades dentro deles.

Conforme definido pelo ITIL (2007), um serviço é "um meio de entregar valor aos clientes, facilitando os resultados que os clientes desejam alcançar sem a propriedade de custos e riscos específicos". Em outras palavras, os clientes querem alcançar seus objetivos sem precisar se preocupar com os processos, pessoas e problemas envolvidos ao construir esses serviços. O conceito de um serviço, bem como a necessidade de gerenciá-lo, precede a própria Tecnologia da Informação, já que suas origens podem ser atribuídas a empresas tradicionais (ITIL, 2007).

Tal dependência pode causar uma reação negativa, as empresas sabem que se seus sistemas que executam os serviços não funcionam tão bem quanto possível, há uma chance de perder clientes. Com isso, nos últimos anos, muitas empresas adotaram a terceirização como forma de montar suas equipes de TI, buscando provedores de serviços capazes de entregar e suportar aqueles sistemas com maior qualidade, evitando assim problemas como lentidão e longos períodos indisponíveis (Bahli and Rivard, 2003).

Os serviços de TI são construídos em infraestruturas (servidores, HDs, equipamentos de rede, e etc) que são bastante complexos e à medida que o número de serviços fornecidos aumenta, também aumenta a complexidade da rede (Johnson et al., 2007). Se os serviços não podem ser interrompidos, a infraestrutura de suporte deve ser gerenciada de perto, mudanças e atualizações devem ser planejadas para que sejam realizadas de maneira ordenada, reduzindo os riscos. Além disso, quando ocorrem falhas, elas precisam ser registradas, analisadas e tratadas com eficiência e atenção (Johnson et al., 2007). Com isso é dada a necessidade de práticas de Gerenciamento de Serviços de Tecnologia da Informação (*ITSM - Information Technology Service Management*).

O *ITSM* é uma abordagem para operações de TI que enfatiza os seus serviços, clientes, a qualidade dos serviços e o manuseio das atividades diárias da função de TI por meio de processos claros, consequentemente, a gerencia de TI como um serviço (Iden and Eikebrokk, 2013). A definição para a estrutura geral que define o Gerenciamento de Serviços de Tecnologia da Informação vem do padrão internacional ISO / IEC 20000 (Van Selm, 1970).

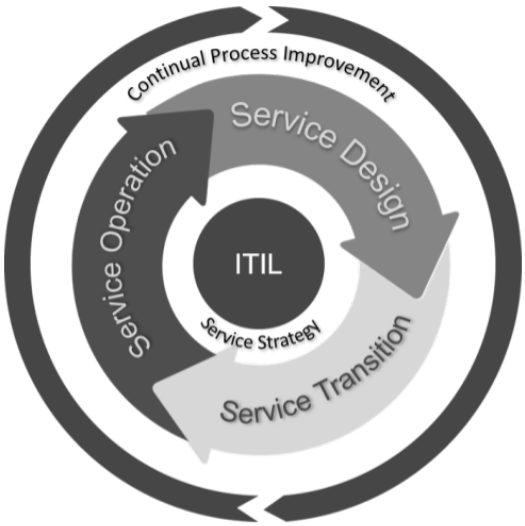

Figura 2.1 – Ciclo de vida ITIL (ITIL, 2007)

Na gestão dos recursos tecnológicos de uma empresa, os incidentes são definidos como eventos inesperados que podem fazer com que os serviços que são suportados por ela sofram perdas de qualidade (ITIL, 2007). Por exemplo, a carga máxima suportada por um sistema é qualificada como um incidente, mesmo que não esteja afetando o desempenho do serviço no momento.

O objetivo do processo de gerenciamento de incidentes é lidar com esses tipos de eventos o mais rápido possível, a fim de minimizar os efeitos negativos, ou evitá-los completamente (ITIL, 2007). Cumprindo assim o *SLA* (Acordo de Nível de Serviço), que especifica os padrões de qualidade e o tempo de disponibilidade que um serviço deve ter (ITIL, 2007).

Existem muitas maneiras diferentes de estruturar os *Service Desks* para executar as diversas etapas do gerenciamento de incidentes, por isso é natural que as organizações construam essa função da maneira que melhor os atendam. Os *Service Desks* podem ser (ITIL, 2007):

- Locais - quando estão posicionados no mesmo ambiente da infraestrutura pelos quais são responsáveis;
- Centralizados - quando todos os seus profissionais estão localizados no mesmo local, às vezes longe dos usuários que estão relatando os incidentes;
- Virtuais - quando *Service Desks* distribuídos geograficamente aparecem para os usuários como uma única unidade;
- *"Follow the Sun"* - quando são estruturados de forma que existam unidades em todo o mundo, com incidentes sendo encaminhados para países que estão no horário de trabalho, garantindo uma cobertura de 24 horas de atendimento.

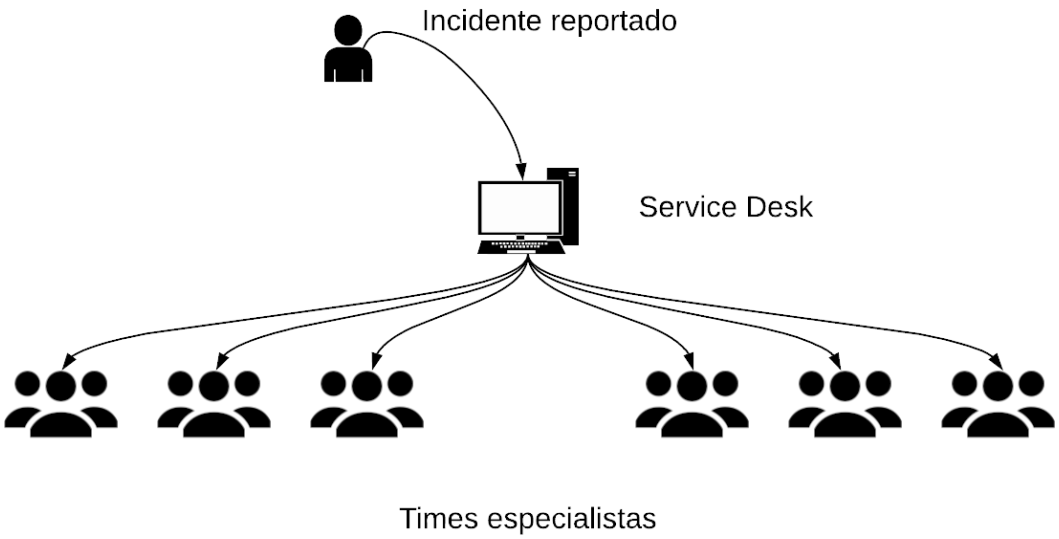

Figura 2.2 – Processo manual de encaminhamento de *tickets*

O gerenciamento de incidentes pode ser iniciado tanto com uma reclamação de um usuário ou mesmo do pessoal técnico da empresa e etc. Uma vez que os incidentes são apontados, os seguintes procedimentos são seguidos (ITIL, 2007):

- Identificação do incidente - Como os incidentes precisam ser resolvidos antes que impactem os serviços, um forte processo de monitoramento deve ser implementado para que os incidentes sejam identificados rapidamente. Sem identificação, os incidentes não podem ser tratados.
- Registro do incidente - Informações essenciais que precisam ser registradas incluem descrição, sintomas, impacto, prioridade, entre outros. Nesta etapa, o *ticket* de incidente é criado.

- Categorização do incidente - Os incidentes são categorizados para manter um certo nível de controle sobre os tipos de incidentes que estão ocorrendo, além de facilitar o encaminhamento para as equipes especializadas. Após essa etapa, se os incidentes relatados forem rotulados como meras solicitações de serviço, que não têm o potencial de interromper os serviços, eles serão removidos do fluxo de gerenciamento de incidentes e tratados separadamente.

- Priorização do incidente - A atribuição de uma prioridade a um incidente pode determinar como ele será tratado pelas ferramentas e pela equipe de suporte. Em geral, a prioridade do incidente é calculada levando em conta seu impacto e urgência. Se os incidentes forem considerados críticos, é possível tratá-los separadamente através de um fluxo exclusivo.

- Investigação e diagnóstico - Durante a etapa de diagnóstico, o analista que lida com o incidente pode se aprofundar ainda mais no problema para obter mais detalhes sobre o que está acontecendo, de modo que essas informações sejam adicionadas ao *ticket* de incidente, que futuramente pode ajudar na solução. Após o diagnóstico, se o analista sentir que não consegue resolvê-lo, poderá encaminhá-lo para outras equipes mais especializadas.

- Resolução e recuperação - Uma resolução é identificada, aplicada e testada, para depois recuperar o estado normal do serviço. É importante que as informações referentes a esses procedimentos sejam registradas no *ticket* para que um histórico completo dos incidentes seja mantido.

- Fechamento do incidente - Depois de verificar que a solução aplicada realmente resolveu o incidente e que os usuários estão satisfeitos com os resultados, o *ticket* é fechado. Nesta etapa, os profissionais que lidam com o *ticket* também podem verificar se a categorização foi feita corretamente e certificar-se de que todo o histórico do incidente foi registrado.

2.2 Modelagem de Processos

No começo usada principalmente para fluxos de processos de fabricação, a modelagem de processos vem sendo amplamente utilizada. Na década de 1960, usada para programação de computadores, na década de 1980, começou a ser usada para descrever um processo de negócio. Ele realmente se consolidou em meados da década de 1980 com o surgimento do livro chamado *Reengineering the Corporation: A Manifesto for Business Revolution* dos autores Hammer and Champy (2009).

Na década de 1980, as organizações começaram a mostrar seu interesse por processos (Rummler and Brache, 2012). Os registros (grandes volumes de dados) arma-

zenados nos sistemas internos tornaram-se importantes fontes. Esses dados incluem *logs* de eventos que podem ser utilizados na mineração de processos. Um *log* de evento inclui registros específicos sobre quando e por quem as atividades de negócios foram realizadas (Park and Kang, 2016).

E então veio a revolução da Internet e a noção de *E-business* e *E-commerce* nos anos 90. Essa revolução na maneira de pensar sobre a criação de empresas virtuais, exigiu uma nova maneira de modelar seus negócios, em outras palavras, surgiu uma necessidade mais forte de modelagem de processos de negócios. Isso recentemente se concretizou com a publicação da Linguagem e Notação de Modelagem de Processos de Negócios (BPML e BPMN) pela *Business Process Management Initiative* (Park and Kang, 2016).

Os Sistemas de Informação estão cada vez mais entrelaçados com os processos operacionais que eles suportam (Van Der Aalst, 2011). Como resultado, uma infinidade de eventos é registrada pelos sistemas de hoje. No entanto, as organizações têm problemas para extrair valor desses dados. O objetivo da mineração de processos é usar dados de eventos para extrair informações relacionadas aos processos, por exemplo, para descobrir automaticamente um modelo observando eventos registrados por algum sistema (Van Der Aalst, 2011).

Em outras palavras o objetivo da análise de mineração de processos é melhorar e não unicamente analisar os dados relacionados à execução dos processos. Em geral, quando um projeto de mineração de processos é iniciado, questões ou problemas centrais que devem ser respondidos são definidos. Então, a análise de mineração de processo é realizada para encontrar uma resposta a essas perguntas ou problemas (Mans et al., 2015).

Com foco em estudar, documentar e melhorar os processos do *Service Desk* de uma empresa. O *Business process modeling (BPM)* ou modelagem de processos de negócios é normalmente realizado por analistas de negócios com suas expertises, por especialistas no assunto que possuem conhecimento especializado dos processos que estão sendo modelados ou mais comumente por uma equipe que compreende ambos (Dufresne and Martin, 2003).

Os *BPM*s descrevem como um negócio funciona, ou mais especificamente, como eles realizam missões, atividades ou tarefas. Um único modelo mostra como uma empresa realizou uma única tarefa, seriam necessários muitos modelos de processos para detalhar como realmente a empresa funciona (Dufresne and Martin, 2003).

Modelos de processos já vinham sendo utilizados mesmo antes do início dos estudos na área. Mesmo que básicos, esses modelos eram usados para treinar novos funcionários, auxiliar nos processos de reengenharia, usados para desenvolver simulações para testar os processos em entradas que não ocorreram na vida real. Finalmente, eles podem ser usados para desenvolver sistemas para automatizar os

processos que eles modelam (Dufresne and Martin, 2003). No último caso, os programadores podem usar o modelo de processo como um guia no desenvolvimento do Sistema de Informações ou, mais recentemente, alguns modelos de processo podem ser executados através de mecanismos de execução de processos que automatizam o processo diretamente a partir do modelo (Dufresne and Martin, 2003).

Além de ter muitos usos, os modelos de processos podem ser criados ou apresentados usando muitas metodologias diferentes. Cada metodologia tem um histórico diferente, muitos têm uma maneira diferente de analisar os processos com base no propósito para o qual foram originalmente criados. Alguns são mais ou menos legíveis por não profissionais. Alguns são mais ou menos úteis para modelagem de processos de negócios (Van Der Aalst, 2011).

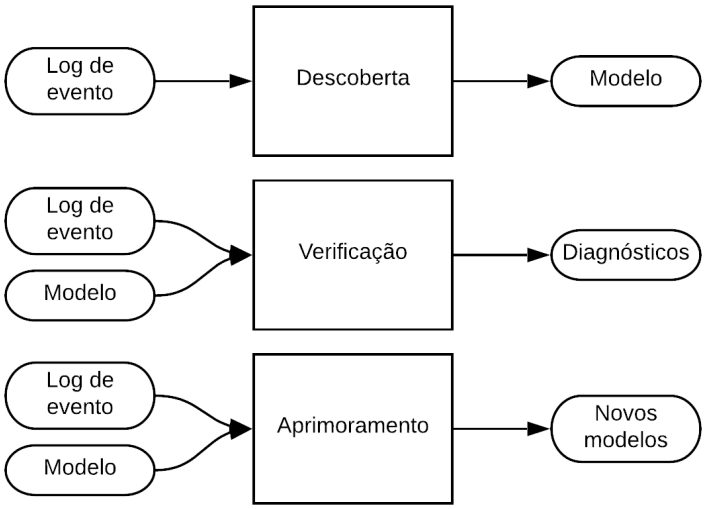

Figura 2.3 – Três tipos básicos de mineração de processos (Van Der Aalst et al., 2011)

A figura 2.3 descreve os três tipos de mineração de processos em termos de entrada e saída. As técnicas de descoberta usam um *log* de eventos e produzem um modelo, que pode ser por exemplo uma *Petri Net* ou *BPMN*. As técnicas de verificação de conformidade precisam de um *log* de eventos e um modelo como entrada e a saída consiste em informações de diagnósticos mostrando diferenças e semelhanças entre modelo e *log*. Técnicas para aprimoramento de modelos também precisam de um *log* de eventos e um modelo como entrada, a saída é um modelo aprimorado ou estendido (Van Der Aalst et al., 2011).

2.2.1 Fluxogramas (*Flow Charts*)

Foram adotados por programadores para descrever a lógica de execução de um programa. Tem número reduzido de símbolos e pouco poder semântico para representar processos complexos, se considerarmos todas as várias formas de fluxogramas, eles são um dos mais comuns do planeta, usados por pessoas técnicas e não técnicas em vários campos (Dufresne and Martin, 2003).

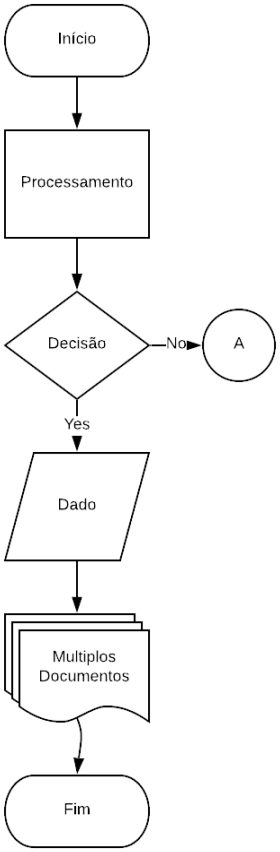

Figura 2.4 – Fluxograma

2.2.2 Diagramas de fluxos de dados (*DFD - Data Flow Diagrams*)

Os *DFDs* têm seu nascimento em 1978. Como pode ser inferido a partir do nome, concentram-se nos dados de um sistema de informação, mostrando a sequência de etapas de processamento percorridas pelos dados. Cada etapa documenta uma ação realizada para transformar ou distribuir os dados. Os *DFDs* são fáceis de ler, possibilitando que especialistas em domínio criem e validem os diagramas (Dufresne and Martin, 2003).

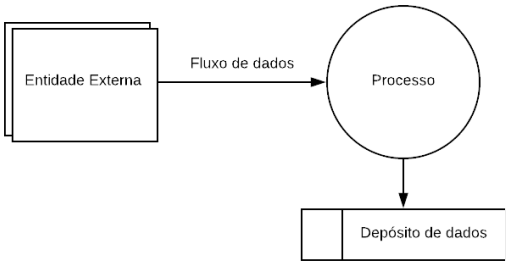

Figura 2.5 – Diagrama de fluxo de dados

2.2.3 Diagrama de fluxo de controle (*CFD - Control Flow Diagrams*)

CFDs são semelhantes aos *DFDs*, porém podem ser usados para aplicações orientadas a eventos. A abordagem *DFD* é baseada em técnicas clássicas de análise estruturada e teoria da máquina de estados finitos e une os dois em uma nova abordagem. As máquinas de estado finito são usadas para controlar o comportamento dos diagramas de fluxo de dados complementares (*DFD*) (Dufresne and Martin, 2003).

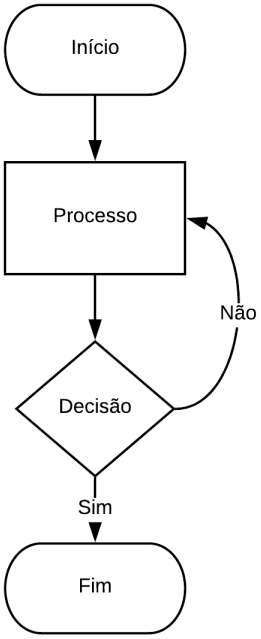

Figura 2.6 – Diagrama de fluxo de controle

2.2.4 Notação de Modelagem de Processos de Negócios (*BPMN - Business Process Modeling Notation*)

BPMN é uma técnica relativamente nova para o desenvolvimento de modelos de processos. Foi publicado pela Iniciativa de Gerenciamento de Processos de Negócios (*Business Process Management Initiative - BPMI*) como uma especificação preliminar para modelagem de processos de negócios. O *BPMN* tem uma linguagem formal baseada em *XML* correspondente chamada *BPML (Business Process Modeling Language)* (Dufresne and Martin, 2003).

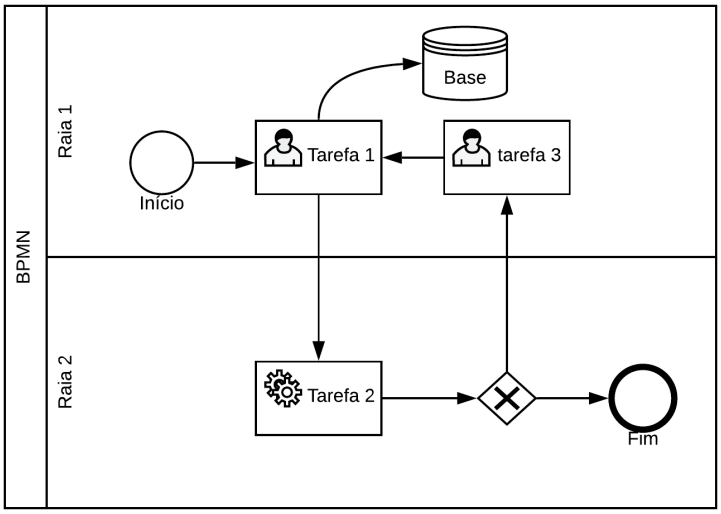

Figura 2.7 – Notação de modelagem de processos de negócios

2.3 Técnicas utilizadas

2.3.1 Processamento de Texto

A transformação de texto é a conversão do conteúdo de um documento textual para que possa ser reconhecido por um computador, permitindo que a máquina o processe e classifique (Lan et al., 2008). Nos experimentos que foram feitos para investigar as melhores maneiras de categorizar *tickets* de incidentes, o texto foi transformado em vetores numéricos por modelos de espaço vetorial, onde cada dimensão do vetor corresponde a um termo no texto.

2.3.2 Termo Frequência

Técnica amplamente utilizada na área de Recuperação de Informação (*Information Retrieval*) afim de representar documentos textuais como um vetor que contém valores numéricos para quantificar as palavras e termos contidos nesse documento (Roelleke, 2013).

Exemplo na aplicação dos *tickets*:

1 - "Não consigo acessar o sistema."

2 - "O sistema não está respondendo, não consigo acessar."

***Ticket* ID**	**não**	**consigo**	**acessar**	**o**	**sistema**	**está**	**respondendo**
1	1	1	1	1	1	0	0
2	2	1	1	1	1	1	1

Tabela 2.1 – Vetor que representa os *tickets*

O método *TF-IDF* leva em consideração a frequência com que o termo está no restante dos documentos. Termos que costumam ser frequentes em todo o conjunto de documentos, como artigos e preposições, são ponderados para diminuir sua importância (Ramos et al., 2003).

$$TF_{ij} = \frac{|frequencia\ do\ termo\ i\ no\ documento\ j|}{|total\ de\ palavras\ no\ documento\ j|} \tag{2.1}$$

$$IDF_i = \log \frac{|total\ de\ documentos|}{|documentos\ com\ termo\ i|} \tag{2.2}$$

2.3.3 Classificadores

A classificação de texto é a tarefa de classificar automaticamente os documentos de linguagem natural que não são rotulados (ou seja, não é conhecido a qual classe eles pertencem) em um conjunto predefinido de categorias (Lan et al., 2008).

A classificação é uma questão fundamental no aprendizado de máquina e na mineração de dados. Na classificação, o objetivo de um algoritmo de aprendizado é construir um classificador dado um conjunto de exemplos de treinamento com rótulos de classe. Normalmente, um exemplo E é representado por uma tupla de valores de atributos $(x_1, x_2, \ldots, x_n)$, onde x_i é o valor do atributo X_i (Zhang, 2004). Após o texto dos documentos ter sido transformado, a classificação do texto é feita alimentando esses dados em algoritmos de aprendizagem de máquina.

2.3.4 *Naive Bayes*

Os métodos *Naive Bayes* são um conjunto de algoritmos de aprendizado supervisionados baseados na aplicação do teorema de Bayes, mais simples, assume que todos os atributos são independentes, dado o valor da variável de classe, chamado de independência condicional. Mesmo essa independência condicional sendo raramente verdadeira em cenários reais (Zhang, 2004).

O teorema de Bayes declara a seguinte relação, dada a variável de classe y e o vetor de recurso dependente x_1 através de x_n (Zhang, 2004):

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Usando a suposição da independência condicional ingênua de que:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Como $P(x_1, \ldots, x_n)$ é constante dada a entrada, podemos usar a seguinte regra de classificação:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Apesar de suas suposições aparentemente simples, os classificadores *Naive Bayes* têm funcionado bem em muitas situações do mundo real, como por exemplo em classificação de documentos e filtragem de spam. Eles exigem uma pequena quantidade de dados de treinamento para estimar os parâmetros necessários (Zhang, 2004).

Naive Bayes podem ser extremamente rápidos em comparação com métodos mais sofisticados. O desacoplamento das distribuições de características condicionais de classe significa que cada distribuição pode ser estimada independentemente como

uma distribuição unidimensional. Isso, por sua vez, ajuda a aliviar os problemas decorrentes da maldição da dimensionalidade (Zhang, 2004).

2.3.5 *K-Nearest Neighbors*

A classificação baseada em vizinhos é um tipo de aprendizado baseado em instâncias ou de aprendizado não generalizante: ela não tenta construir um modelo geral, mas armazena instâncias dos dados de treinamento. A classificação é calculada a partir de uma votação por maioria simples dos vizinhos mais próximos de cada ponto, ou seja, um ponto de consulta é atribuído à classe de dados que possui mais representantes dentro dos vizinhos mais próximos do ponto (Weinberger et al., 2006).

A escolha ideal do valor k é altamente dependente dos dados, em geral, um k maior suprime os efeitos do ruído, mas torna os limites de classificação menos distintos (Weinberger et al., 2006).

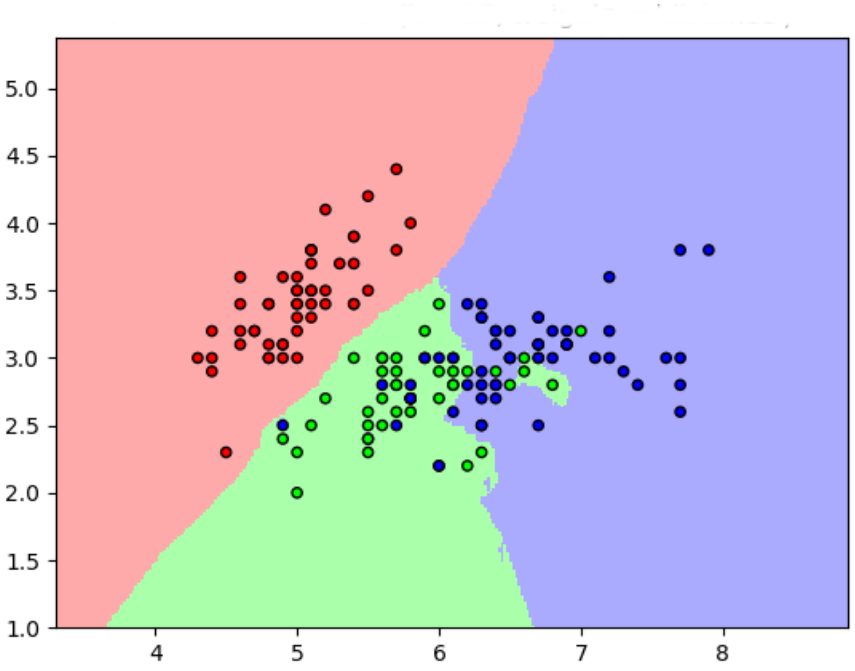

Figura 2.8 – Exemplo de classificação de 3 classes com *Knn* (Pedregosa et al., 2011)

2.3.6 *Support Vector Machine (SVM)*

Uma Máquina de Vetores de Suporte constrói um hiperplano ou conjunto de hiperplanos em um espaço de alta dimensão (tendendo ao infinito), que pode ser usado para classificação, regressão ou outras tarefas. Intuitivamente, uma boa separação é alcançada pelo hiperplano que possui a maior distância até os pontos de dados

de treinamento mais próximos de qualquer classe (a chamada margem funcional), pois em geral quanto maior a margem menor o erro de generalização do classificador (Guyon et al., 1993).

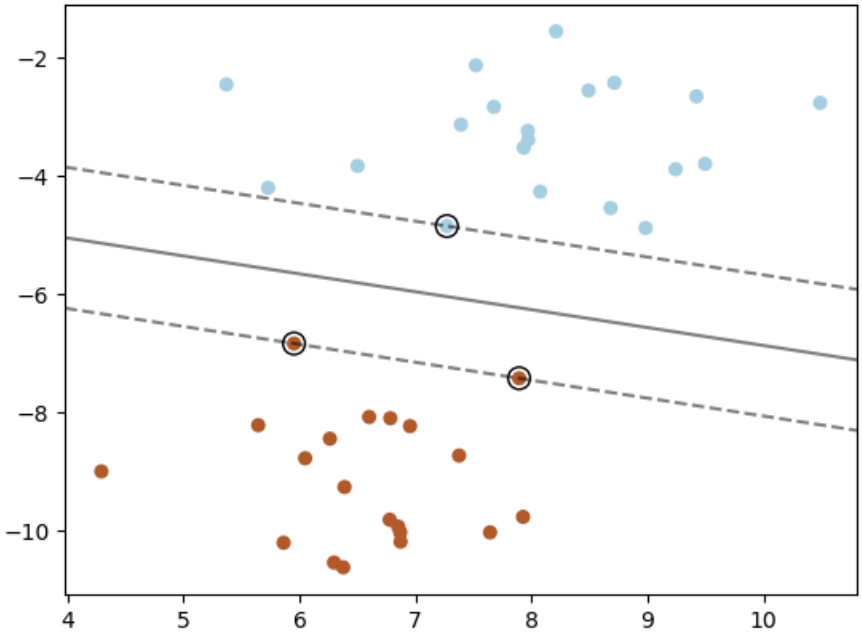

Figura 2.9 – Exemplo *SVM* (Pedregosa et al., 2011)

Dados vetores de treinamento, $x_i \in \mathbb{R}^p, i = n, \ldots, n$, em duas classes e um vetor $y \in \{1, -1\}^n$, o *SVM* para classificação resolve o seguinte problema primordial (Guyon et al., 1993):

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{2.3}$$

$$\text{sujeito à} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \tag{2.4}$$

$$\zeta_i \geq 0, i = 1, ..., n \tag{2.5}$$

Simplificando:

$$\min_\alpha \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{2.6}$$

$$\text{sujeito à} \quad y^T\alpha = 0 \tag{2.7}$$

$$0 \leq \alpha_i \leq C, i = 1, ..., n \tag{2.8}$$

Onde e é um vetor de somente valores unitários, $C > 0$ é o limite superior, Q é uma matriz nxn semi definida positiva, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, onde $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ é o *kernel*. Aqui vetores de treinamento são mapeados implicitamente em um espaço dimensional superior (talvez infinito) pela função ϕ (Guyon et al., 1993).

A função de decisão é:

$$\text{sgn}(\sum_{i=1}^{n} y_i\alpha_i K(x_i, x) + \rho)$$

As vantagens das *SVMs* são (Guyon et al., 1993):

- Eficaz em espaços dimensionais elevados;
- Ainda é eficaz nos casos em que o número de dimensões é maior que o número de amostras;
- Diferentes funções de *Kernel* podem ser especificadas para a função de decisão.

As desvantagens incluem (Guyon et al., 1993):

- Se o número de variáveis for muito maior que o número de amostras, é necessário maior atenção nos ajustes dos parâmetros.

2.3.7 Base de Dados Desbalanceadas

O problema de classes desbalanceadas vem atraindo grande interesse de estudos e muitos métodos foram desenvolvidos para lidar com esse problema (Wang and Yao, 2012). Classes desbalanceadas acontecem quando uma classe é altamente representada na base de dados quando comparada com outras, essa distribuição pode fazer com que os classificadores se tornem tendenciosos, fazendo previsões precisas para amostras da classe majoritária e falhando em obter resultados bons o suficiente para as minoritárias (Wang and Yao, 2012).

Devido à natureza generalizada do problema, muitas abordagens para lidar com classes desbalanceadas foram propostas. Eles podem ser agrupados em três categorias (FernáNdez et al., 2013):

- Soluções à Nível de Dados - O objetivo é eliminar amostras da classe majoritária ou criar/replicar amostras da classe minoritária. Como consequência, um conjunto de dados originalmente desequilibrado pode ser artificialmente criado para ser mais equilibrado;

- Soluções de Nível Algorítmico - A adaptação não é realizada nos dados, mas nos algoritmos de classificação. Eles são alterados para dar mais foco as classes minoritárias.

- Soluções Sensíveis a Custos - Os algoritmos são forçados a prestar mais atenção às classes minoritárias, custos mais altos de classificação incorreta são aplicados às suas amostras.

As soluções à nível de dados podem ser divididas em:

- *Oversampling* - As classes menos representadas terão seu número de exemplos aumentados a partir de criações sintéticas de novos exemplos a partir dos existentes Chawla et al. (2002).

- *Undersampling* - Na contramão do ponto anterior, o foco é podar as classes que tem mais exemplos até o ponto de todas ficarem em mesma quantidade Zhang et al. (2010).

O *Random Oversampling* é a abordagem mais direta, seleciona aleatoriamente amostras das classes minoritárias e as replica. A replicação é feita até que o número de itens nas classes minoritárias seja igual ao número da classe majoritária. Existe também a versão *Undersampling* que ao contrário da versão anterior elimina amostras aleatoriamente das classes majoritárias até que todas as classes tenham o mesmo números de exemplos (Chawla, 2009).

O *SMOTE* (Chawla et al., 2002) cria amostras sintéticas selecionando pontos da classe minoritária, identificando os k vizinhos mais próximos daquele ponto que também são da mesma classe e criando um novo exemplo em uma parte aleatória do segmento de linha que conecta dois pontos de dados. O valor k é calculado de acordo com a taxa de super amostragem que precisa ser feita para que as classes minoritárias tenham a mesma quantidade de amostras que a classe majoritária.

A técnica de *Undersampling Cluster Centroids* (Zhang et al., 2010), submete as amostras substituindo um *cluster* das classes majoritárias pelo seus centroides de um algoritmo *K-Means*. O algoritmo é repetido n vezes até que todas as classes tenham o mesmo número de exemplos da classe minoritária.

2.3.8 Redução de Dimensionalidade

Muitos problemas no processamento de sinais e imagens, aprendizado de máquina e reconhecimento de padrões exigem lidar com problemas de dimensões muito altos nos conjuntos de dados (Shahid et al., 2016). É comum em técnicas de representação textuais, como os vetores formados pela frequência das palavras do documento, sofrer do problema da alta dimensionalidade. O vetor resultante terá tantas dimensões quantas forem as palavras no vocabulário (todas as palavras dos textos da base de dados excluindo repetições). Existem técnicas que se propõe a diminuir as dimensões desse vetor sem perda significativa de informação. Como é o caso da técnica de Análise de Componentes Principais (*Principal Component Analysis - PCA*) (Shlens, 2014).

O *PCA* utiliza uma transformação para converter as variáveis originais em outro conjunto de mesma quantidade de variáveis independentes entre si chamadas de componentes principais. As componentes principais são calculadas com o propósito de guardar o máximo de informação dos dados originais, de forma que a primeira componente principal é a que melhor representa os dados (Shlens, 2014).

Capítulo 3

Proposta

3.1 Trabalhos Relacionados

Devido a importância de manter os Serviços de TI o máximo de tempo em funcionamento, evitando a perda de qualidade e com isso a insatisfação dos seus clientes, muitos estudos vem sido desenvolvidos para auxiliar no tratamento de incidentes. Seja identificando pontos críticos dentro dos processos da empresa ou mesmo automatizando decisões que normalmente são feitas por pessoas.

O trabalho (Moharreri et al., 2016) busca medir a qualidade do fornecimento de serviços. Nessa pesquisa, é utilizado a métrica de Tempo Médio Para Resolução (*Mean Time To Resolve - MTTR*), que avalia o tempo médio necessário para que os incidentes sejam resolvidos após serem relatados. Para atingir esse objetivo, a pesquisa trata do problema de encaminhar *tickets* de incidentes por meio de uma Rede Coletiva de Especialistas (*Collective Expert Network - CEN*), ou seja, procura uma maneira de recomendar o caminho apropriado de especialistas que devem tratar cada *ticket*. Para isso, primeiro é determinado os caminhos mais comuns entre os grupos de especialistas (utilizando técnicas de Mineração de Processos), depois são usadas técnicas de processamento de texto e aprendizado de máquina para, primeiro transformar a descrição do *ticket* em um vetor e em seguida, determinar qual dos caminhos será recomendado para solucionar *ticket*.

O trabalho (Kikuchi, 2015) tenta identificar a carga de trabalho necessária para tratar cada *ticket* de acordo com o histórico de atualizações dos incidentes semelhantes. A carga de trabalho de cada *ticket* é estimada através do seu histórico e os que já foram resolvidos são rotulados com uma das duas categorias, fácil ou difícil, a depender do número de registros no histórico. O texto dos *tickets* é transformado com *TF-IDF* e o algoritmo *Naive Bayes* é utilizado para aprender as duas classes. Com as cargas de trabalho dos *tickets* estimadas automaticamente, é possível gerenciá-las

com mais clareza, além do esforço despendido na análise do *ticket* e na tentativa de determinar o quão difícil será resolvê-lo é eliminado.

O artigo (Zhou et al., 2015) parte do ponto que já que muitos *tickets* tem soluções parecidas ou iguais, e desenvolve uma estrutura utilizando processamento de texto e aprendizado de máquina para recomendar soluções para novos *tickets* recebidos. Dado que palavras diferentes são usadas para descrever problemas semelhantes, também foi utilizado um algoritmo de adaptação de características para identificar essas palavras e considerá-las como uma única.

A tese (Menezes, 2009) avaliou diversas técnicas de transformação e classificação de texto para que as melhores pudessem ser escolhidas para encaminhar *tickets* por meio de um *Service Desk* de três níveis. A tese relatou uma diminuição na carga total de trabalho de 85%. E por último a tese (Ferreira, 2017) que trabalhou com um *Service Desk* de dois níveis avaliando além de algoritmos de classificação de texto, problemas relacionados a registros que variam ao longo do tempo (*Concept Drift*).

3.2 Estudo de Caso

Os *tickets* analisados nesse trabalho foram coletados da base de dados de uma empresa brasileira que implementa o *ITIL* em sua organização. No cenário da empresa, uma única equipe centraliza a criação dos *tickets* e realiza seu encaminhamento para as áreas que possuam a capacidade técnica, bem como a responsabilidade de resolvê-los.

Um *ticket* catalogado na base de dados é formado pelos campos:

1. número da ocorrência - O número único que identifica o *ticket* no sistema;
2. descrição da ocorrência - Campo de texto livre com a descrição do problema dada pelo usuário;
3. data da abertura - Data em que o *ticket* foi aberto no sistema;
4. data do fechamento - Data em que o *ticket* foi encerrado no sistema;
5. problema - Categoria selecionada pelo usuário;
6. *SLA* - Tempo estimado em que ocorrências de uma categoria específica de problema sejam encerradas;
7. *STATUS* - Indica a fase atual do *ticket* no sistema (aberta, encerrada e etc);
8. sistema - Nome do departamento para qual o *ticket* foi enviado.

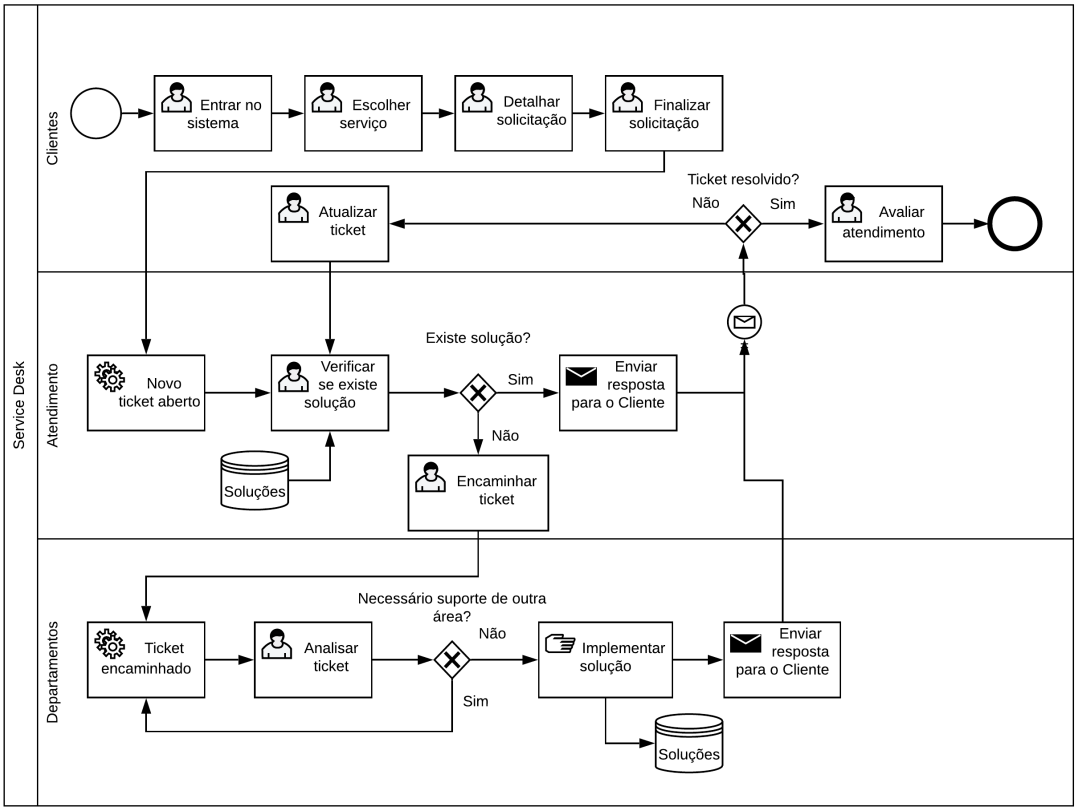

Figura 3.1 – Modelagem de processos de negócios

Esse *ticket* recebido em muitos casos já terá uma solução documentada na base de dados, como mostrado na imagem 3.1, com isso o atendimento pode responder ao cliente com a solução do incidente com base nos casos passados. Caso o *ticket* necessite de atenção especializada ou não tenha uma solução catalogada no sistema, o atendimento repassa o incidente para análise do *Service Desk* para que o possa encaminhar para uma área com expertise para resolver o *ticket*.

É importante notar que esse caminho não é necessariamente dado por um único passo, visto que o *ticket* pode percorrer N áreas até ser resolvido de fato. Inclusive, podendo voltar para o cliente que o criou para que possa introduzir mais informações que possam ajudar na construção da solução.

A base contém um total de 165020 linhas de *logs* de eventos, sendo identificados 63937 instâncias de processos de um total 2851 processos diferentes (caminhos diferentes que o *ticket* percorreu). Os processos tem tamanho mínimo de 1 transição e máximo de 26 transições com uma média de 3 transições entre áreas até o fechamento do *ticket*.

O gerenciamento de incidentes da empresa é formado por 36 grupos de especialistas e o número de ocorrências atendidas pelos times mais acionados pode ser visto na tabela a seguir 3.1.

Times especialistas	*Tickets*	Porcentagem
AGENDAMENTO	42312	25,641%
CCO	38312	23,217%
ADMINISTRAÇÃO/VENDA	22393	13,57%
SUPORTE	21915	13,28%
SUPORTE - CADASTRO	6229	3,775%
CRP	6124	3,711%
LOGISTICA	4301	2,606%
TI	3929	2,381%
SUPORTE - ADMINISTRAÇÃO	2998	1,817%
POS VENDAS	2928	1,774%
Outros	13579	8,229%

Tabela 3.1 – Histograma dos grupos mais acionados

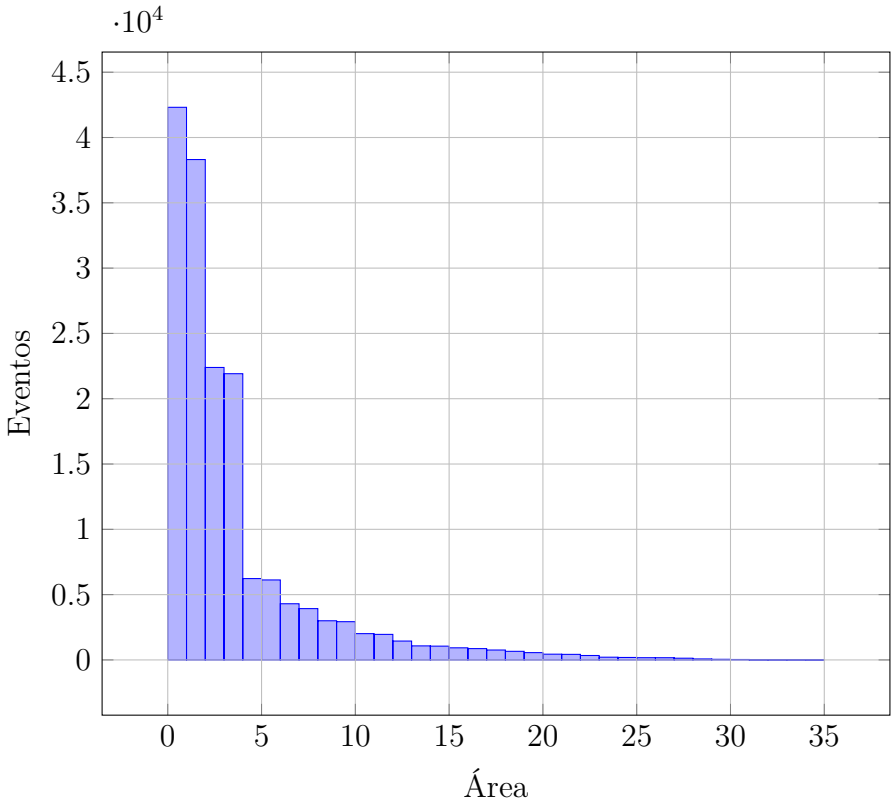

Figura 3.2 – Gráfico dos grupos mais acionados

A tabela 3.1 contabiliza qualquer passagem do *ticket* pelas áreas, ou seja, é contado tanto quando a área resolve o *ticket* quanto quando só tenha tido uma passagem em um processo mais longo para ser resolvido em outra área posteriormente. Já o gráfico 3.2 mostra que mais da metade das áreas são pouquíssimo representadas, o somatório das 14 áreas com menos aparições na base não chega a 1% do total (0,847%).

Podemos destacar também algumas áreas que mais recebem ocorrências no início do processo (seja ela resolvendo o *ticket* em um único passo ou repassando para uma segunda área), vide a tabela 3.2.

Times especialistas	*Tickets*	Porcentagem
SUPORTE	20436	31,963%
ADMINISTRAÇÃO/VENDA	18643	29,158%
CRP	5474	8,562%
SUPORTE - CADASTRO	3613	5,651%
AGENDAMENTO	3010	4,708%
Outros	12761	19,959%

Tabela 3.2 – Histograma das áreas que inciam processos

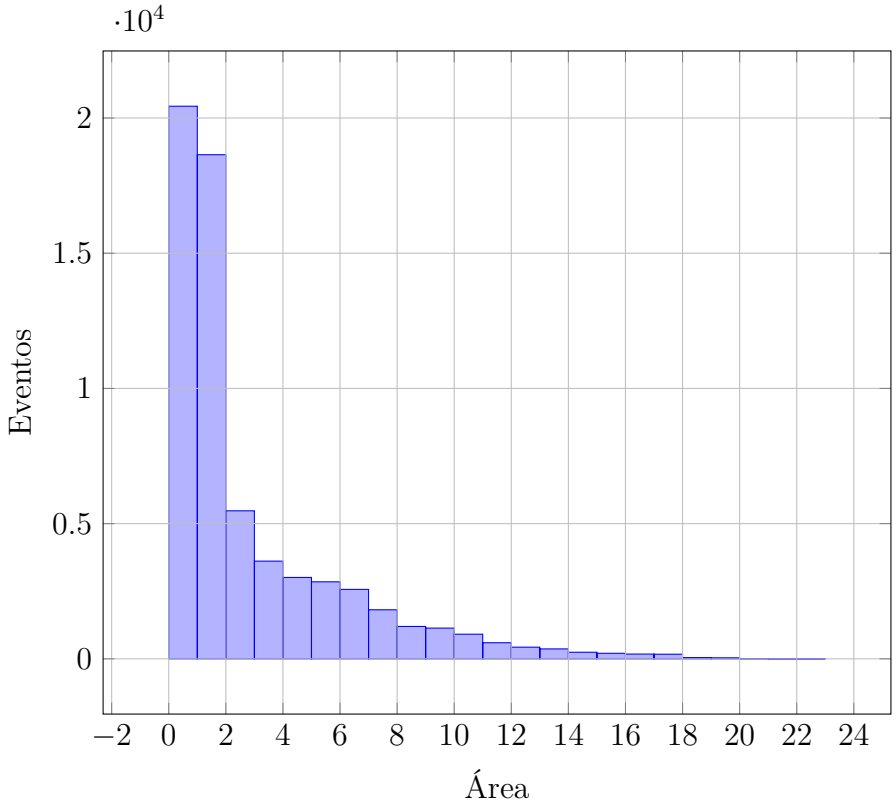

Figura 3.3 – Gráfico dos grupos que iniciam processos

Em uma análise rápida da tabela 3.2 podemos verificar que as classes "SUPORTE" são mais comumente acionadas em um início de processo. Com um total de 24 áreas realizando o papel de dar o *start* no processo, o gráfico 3.3 nos mostra também uma disparidade grande entre os times mais acionados e os menos acionados.

Por sua vez as áreas que mais finalizam atendimentos (novamente, podendo ser a única área dentro do processo de solução ou mesmo a última etapa de um processo mais longo) podem ser vistas na tabela 3.3.

Times especialistas	*Tickets*	Porcentagem
CCO	31063	48,584%
SUPORTE	7423	11,61%
SUPORTE - CADASTRO	5670	8,868%
CRP	4971	7,775%
AGENDAMENTO	3001	4,694%
Outros	11809	18,470%

Tabela 3.3 – Histograma dos grupos que finalizam processos

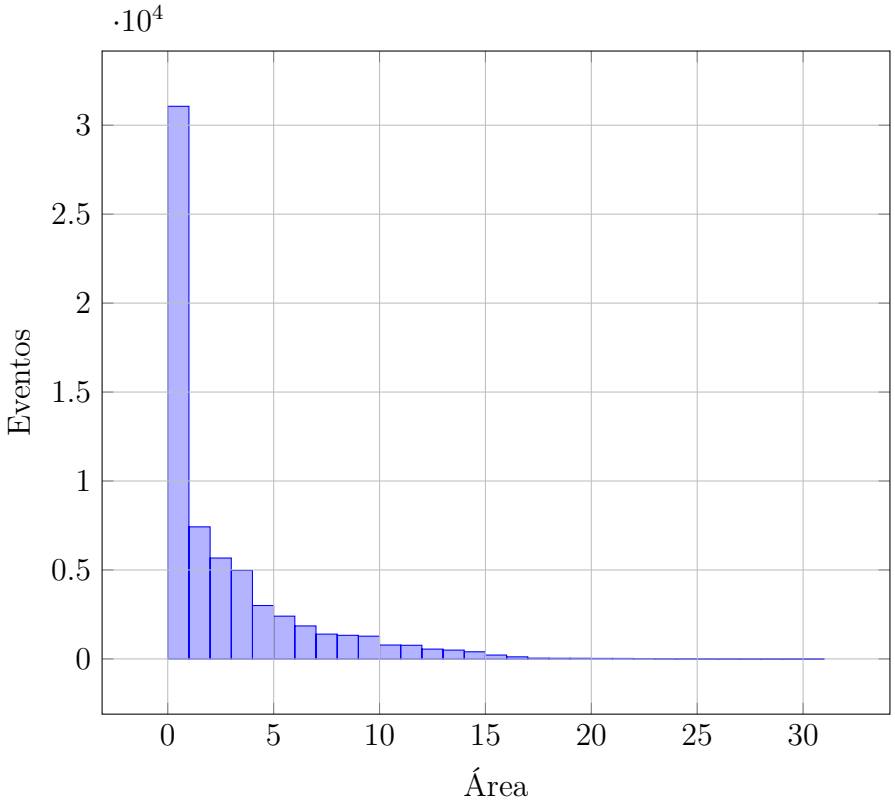

Figura 3.4 – Gráfico dos grupos que finalizam processos

Podemos ver na tabela 3.3 que quase 50% dos *tickets* são finalizados pela área "CCO" e o gráfico 3.4 mostra que a distribuição se mantém muito parecida à dos outros histogramas.

O grafo 3.5 apresenta as transições mais comumente realizadas entre os grupos de especialistas (foram cortadas todas as transições com $\omega' \leq 0.1$). Onde cada nó representa um grupo e cada aresta indica a transição realizada com seu peso indicando a probabilidade condicional da transferência. As auto transições representam os grupos que solucionaram os incidentes.

Sendo os pesos calculados pela formula (Moharreri et al., 2016):

$$\omega' \;=\; P(b \mid a) \;=\; \frac{\omega_{ab}}{\sum\limits_{c \,\in\, Experts \,\wedge\, (a,\, c) \,\in\, Transfers} \omega_{ac}} \tag{3.1}$$

onde ω é a quantidade de *tickets* que executa a transição $(a,\ b)$,

$$\omega \;=\; \mid T_{ab} \mid \tag{3.2}$$

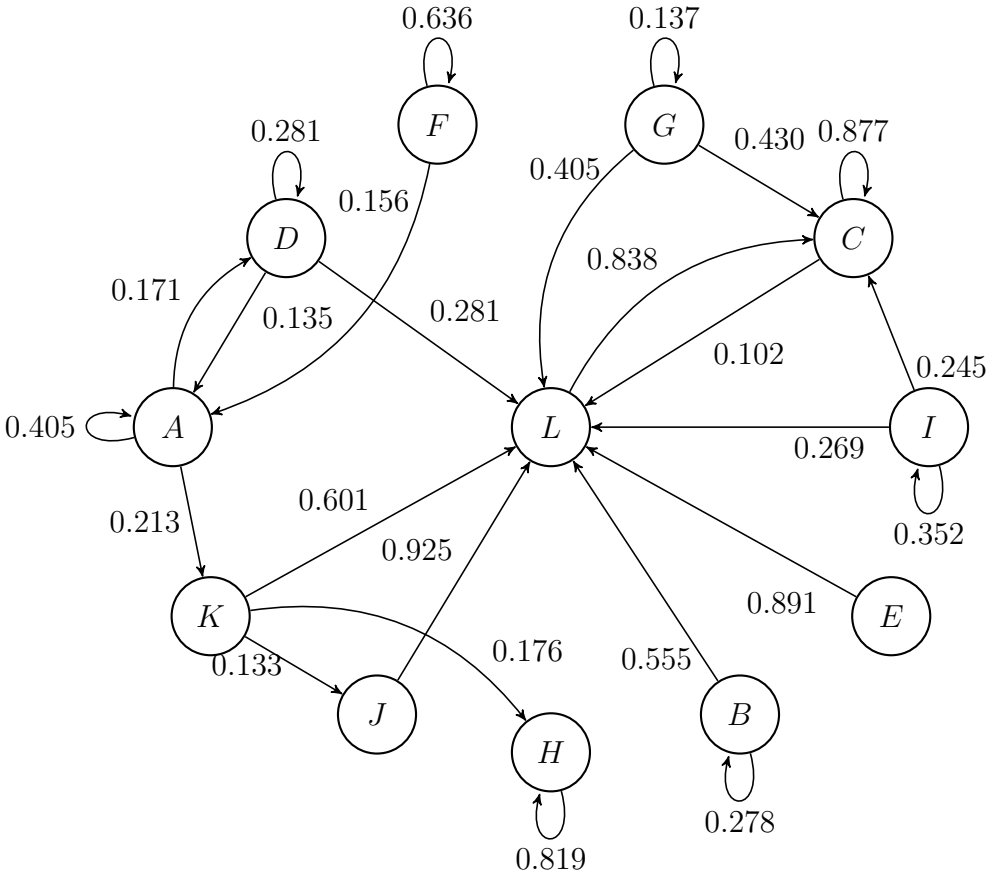

Figura 3.5 – Grafo direcionado com os pesos das transições

- (A) TI
- (B) CRP
- (C) CCO
- (D) POS VENDAS
- (E) SUPORTE - ADMINISTRAÇÃO
- (F) SERVICE DESK
- (G) LOGISTICA
- (H) SUPORTE - CADASTRO
- (I) VAREJO
- (J) ADMINISTRAÇÃO/VENDA
- (K) SUPORTE
- (L) AGENDAMENTO

Podemos ver que o grupo AGENDAMENTO recebe transições de quase todos os outros grupos e que em 80% dos casos aproximadamente os encaminha para o grupo CCO que é o que mais finaliza os *tickets*.

3.3 Metodologia

Nesse trabalho a base de dados foi divida em dois grupos para posteriormente ser submetida aos experimentos. O primeiro grupo contém os incidentes que foram resolvidos com o encaminhamento de apenas um grupo de especialistas, o segundo grupo contém os incidentes que precisaram de mais transições para serem resolvidos.

No primeiro grupo de incidentes foram aplicados os algoritmos de classificação mostrados na sessão 2.3.3. Cada *ticket* foi transformado em um vetor de características *TF-IDF* e rotulado com o identificador do grupo de especialistas que o solucionou.

No segundo grupo de incidentes cada *ticket* também foi rotulado, porém nesse cenário o rótulo é o identificador do caminho percorrido entre os diferentes grupos de especialistas. E a classificação é feita por um modelo baseado no artigo (Moharreri et al., 2016).

O algoritmo *Transformed Weight-normalized Complement Naive Bayes (TWCNB)* (Rennie et al., 2003) é adaptado para utilizar como rótulo um caminho de vários grupos. Onde:

- $\vec{t}$ é o conjunto de *tickets* que foram resolvidos anteriormente: $t = (t_1,\ t_2,\ \ldots,\ t_n)$ e t_{ij} é a frequência da j-ésima palavra do dicionário no *ticket* t_i;
- $\vec{RS} = (\vec{rs}_1,\ \vec{rs}_2,\ \ldots,\ \vec{rs}_n)$ são os caminhos de solução correspondentes a cada um dos *tickets*;
- $C = \{C_1,\ C_2,\ \ldots,\ C_s\}$ é o conjunto de caminhos distintos;
- $\vec{test} = (f_1,\ f_2,\ \ldots,\ f_m)$ é um *ticket* de teste em que f_j é a frequência da j-ésima palavra do dicionário no *ticket* de teste.

Temos:

$$\omega = R - TRS(\vec{t},\ RS) \tag{3.3}$$

$$Predicao(\vec{test}) = \arg\max_{c\ \in\ C} \sum_{j=i}^{m} f_j\ .\ \omega_{jc} \tag{3.4}$$

A função de treinamento é executada pelo algoritmo 1 que transforma cada descrição de *ticket* em vetores *TF-IDF*. ω é a função de normalização ponderada

sobre $P(j|c')$, em que j é o índice de qualquer palavra no dicionário e c é a classe no conjunto de dados (nesse caso, o identificador do caminho que solucionou o *ticket*) (Moharreri et al., 2016).

Algorithm 1: $R - TRS(\vec{t}, \vec{RS})$

for $j = 1$ *até* m **do**
  $IDF_j = \log \frac{n}{\sum_{k=1}^{n} \delta_{kj}}$
  for $i = 1$ *até* n **do**
    $TF_{ij} = \log(t_{ij} + 1)$
for $j = 1$ *até* m **do**
  for $i = 1$ *até* n **do**
    $NC_{ij} = \frac{TF_{ij} \ . \ IDF_j}{\sqrt{\sum_{k=1}^{m} (TF_{ik} \ . \ IDF_k)^2}}$
for $j = 1$ *até* m **do**
  for $h = 1$ *até* s **do**
    $P(j \mid C'_h) = \frac{\lambda + \sum_{k:rs_k \neq c_h}^{n} NC_{kj}}{m\lambda + \sum_{k:rs_k \neq c_h}^{n} \sum_{p=1}^{m} NC_{kp}}$
    $\omega(j \mid C'_h) = \frac{\log P(j \mid C'_h)}{\sum_{k=1}^{m} \log P(k \mid C'_h)}$
return ω

3.3.1 Avaliação

Cada experimento será submetido as métricas *Precision*, *Recall*, *F1-Score*,*Accuracy* e para visualizar o resultado da classificação distribuída entre as diversas classes, será apresentada a matriz de confusão de cada experimento.

- *Precision* - Também conhecida como valor preditivo positivo. É calculado a fração dos itens classificados corretamente com relação ao total de itens classificados de cada classe, para no fim tirar a média (Herlocker et al., 2004):

$$Precision = \frac{|true\ positive|}{|(true\ positive\ +\ false\ positive)|}$$

- *Recall* - Também conhecida como sensibilidade. É calculado a fração dos itens classificados corretamente com relação ao total de itens que deveria ser

classificado de cada classe, para no fim tirar sua média (Herlocker et al., 2004):

$$Recall = \frac{|true\ positive|}{|(true\ positive\ +\ false\ negative)|}$$

- *F1-Score* - Une as medidas anteriores por meio de uma média harmônica (Herlocker et al., 2004).

$$F1Score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

- *Accuracy* - Calcula a precisão global independente da distribuição das classes (Makridakis, 1993).

- Matriz de confusão - Calcula uma matriz para avaliar a precisão de uma classificação. Por definição, uma matriz de confusão C é tal que C_{ij} é igual ao número de amostras na classe i, mas previstas como estando na classe j (Hay, 1988).

Capítulo 4

Experimentos

Os experimentos irão focar apenas na descrição dos *tickets*, para que apenas a partir do texto que o usuário descreveu seja possível realizar o encaminhamento para o grupo correto. Foram utilizados apenas os *tickets* de incidentes catalogados com o STATUS encerrado.

Para a execução dos experimentos foi utilizado uma máquina com processador de 8 núcleos e 56GB de RAM, o programa foi desenvolvido utilizando a linguagem *Python 3.6.8* e usando o *framework sklearn*.

4.1 Classificação de apenas um grupo de especialista

Da base com 165020 linhas de *logs*, 23008 incidentes foram resolvidos após serem encaminhados pelo *Service Desk* para apenas um único grupo de especialistas. Os grupos são formados por 22 times diferentes capazes de resolver o *ticket* e a distribuição de cada um deles na base pode ser vista na tabela 4.1.

4.1.1 Pré-processamento

Foi escolhido eliminar da base as classes com menos de 1% de representatividade, já que por terem poucos exemplos acabariam servindo como ruído na execução dos algoritmos. Com isso os experimentos serão executados com as 12 classes restantes.

Em todos os textos de descrição dos *tickets* foram aplicados transformações para que qualquer pontuação, número e data fossem eliminados. Foram eliminadas também aqueles termos chamados *stopwords*, palavras comuns que carregam pouca importância na identificação do texto como um todo, exemplo artigos, preposições e etc. Foi aplicado também a técnica de *stemming* para que cada palavra fosse substituída por seu radical, diminuindo assim o número de palavras no dicionário.

Classes	Tickets	Porcentagem
Implantação	1	0,004%
Comercial	3	0,013%
Produção	4	0,017%
Qualidade	7	0,030%
AGD FORTEC	23	0,100%
Suporte Credenciado	116	0,504%
Gestão de Contratos	143	0,622%
Agendamento	170	0,739%
Suporte - Administração	203	0,882%
ADM Tecnica	230	1,000%
CCO	237	1,030%
Financeiro	305	1,326%
Engenharia/Produto	311	1,352%
Varejo	373	1,621%
Logistica	755	3,281%
Pos Vendas	787	3,421%
Administração/Venda	1059	4,603%
TI	1710	7,432%
Service Desk	1731	7,523%
Suporte - Cadastro	3325	14,451%
CRP	4648	20,202%
Suporte	6867	29,846%

Tabela 4.1 – Frequência das classes

O dicionário formado por todas as palavras dos textos dos *tickets* após passar pelo pré-processamento descrito acima ficou com um tamanho total de 14927 palavras diferentes. Com a aplicação do *TF-IDF* o resultado é um vetor numérico esparso, que somado a quantidade de *tickets* a serem representados sofrerá com a maldição da dimensionalidade. Para sanar esse problema foi aplicado o algoritmo *PCA* para reduzir a dimensão desses vetores para 20, podendo assim ser carregado em memória. Toda execução é feita utilizando *K-Fold Cross-Validation* com $k = 5$.

4.1.2 Resultados

	Precision	*Recall*	*F1*	*Accuracy*
Naive Bayes	0.470	0.560	0.433	0.510
SVM linear $C = 10$	0.680	0.531	0.559	0.786
Knn $n = 8$	0.707	0.637	0.655	0.805

Tabela 4.2 – Resultados grupo 1

A tabela 4.2 mostra o melhor resultado de cada técnica, as tabelas completas com cada configuração podem ser vistas na sessão A. O algoritmo *Knn* vai melhor

que os demais em todas as métricas, um *F1-Score* de 0.655 e uma acurácia de 0.805 são bons resultados para um problema multi classe, porém a matriz de confusão 4.1 mostra que a última classe (com menos exemplos) não teve nenhuma predição.

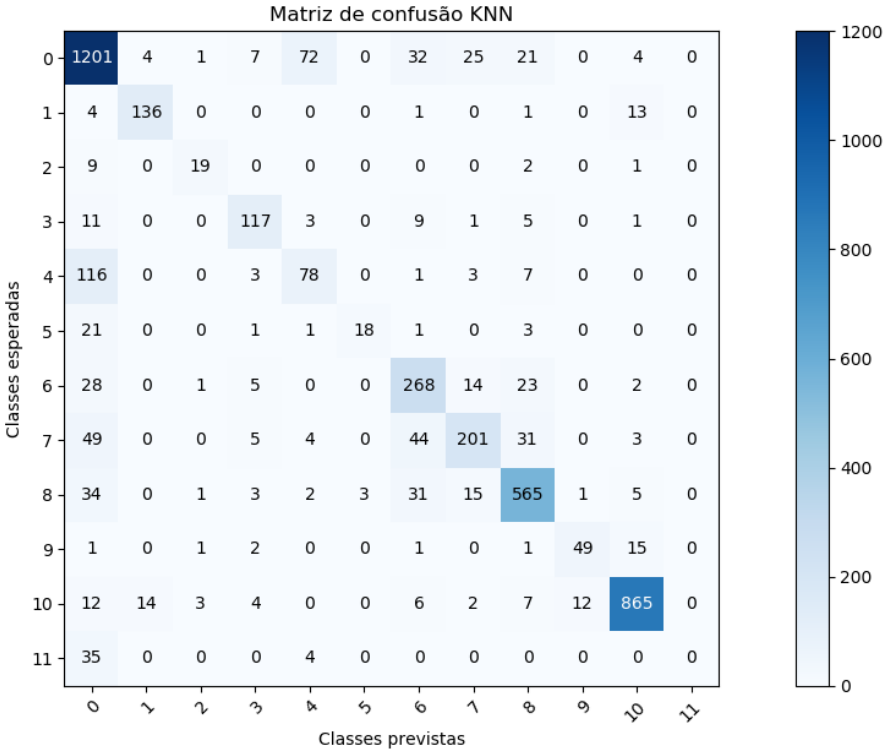

Figura 4.1 – Matriz de confusão do algoritmo *Knn*

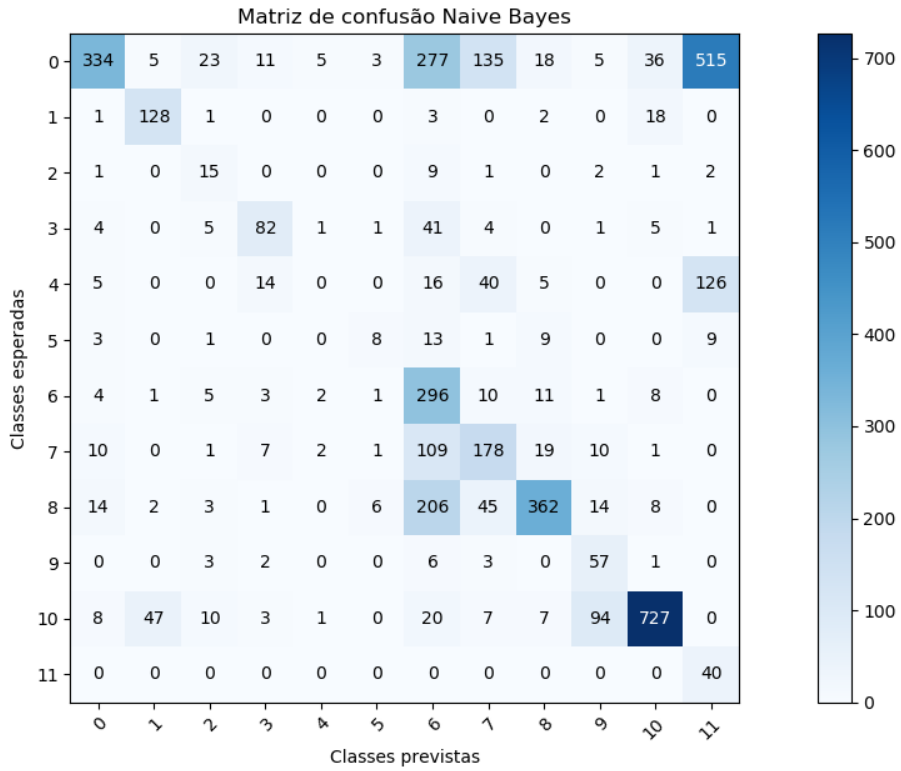

Figura 4.2 – Matriz de confusão do algoritmo *Naive Bayes*

Apesar de mostrar resultados baixos nas métricas utilizadas, o algoritmo *Naive Bayes* foi o único que fez predições para a última classe como pode ser visto na matriz de confusão 4.2.

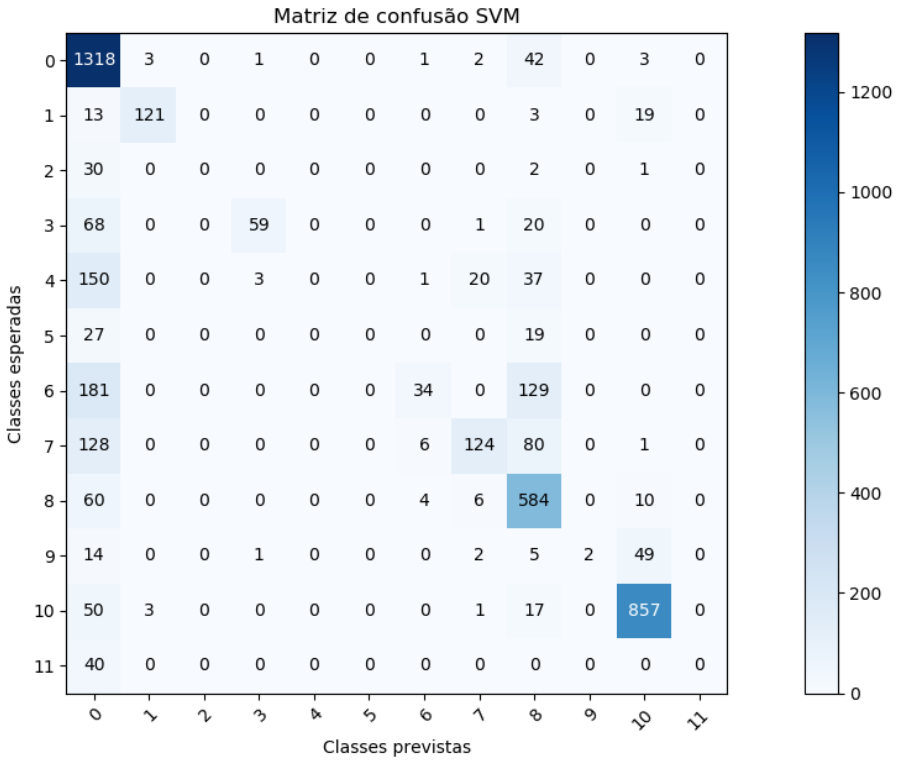

Figura 4.3 – Matriz de confusão do algoritmo *SVM*

Utilizando *SMOTE* para igualar a distribuição das classes dentro da base de dados de treinamento, fazendo com que cada classe tenha um total de 5538 exemplos.

	Precision	*Recall*	*F1*	*Accuracy*
Naive Bayes	0.421	0.554	0.391	0.469
SVM linear $C = 10$	0.552	0.668	0.536	0.619
Knn $n = 2$	0.617	0.672	0.624	0.744

Tabela 4.3 – Resultados grupo 1 com *SMOTE*

A tabela 4.3 mostra que o algoritmo *Knn* continua tendo os melhores resultados em todas as métricas, mas perde sua precisão por fazer predições mais variadas. A imagem 4.4 mostra que todas as classes tiveram valores preditos.

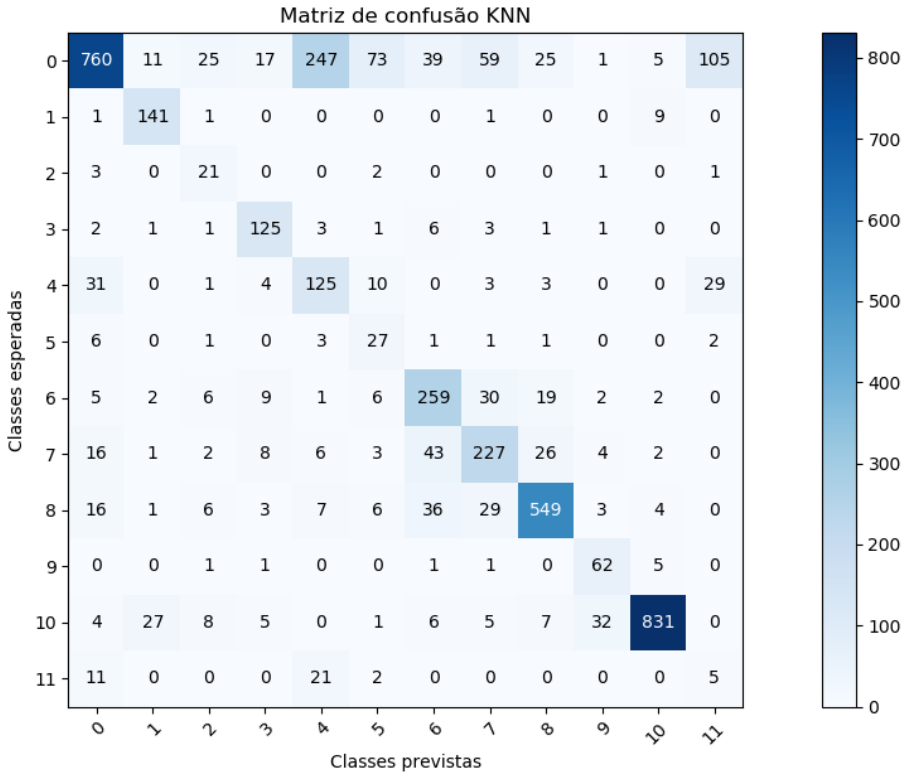

Figura 4.4 – Matriz de confusão do algoritmo *KNN* com *SMOTE*

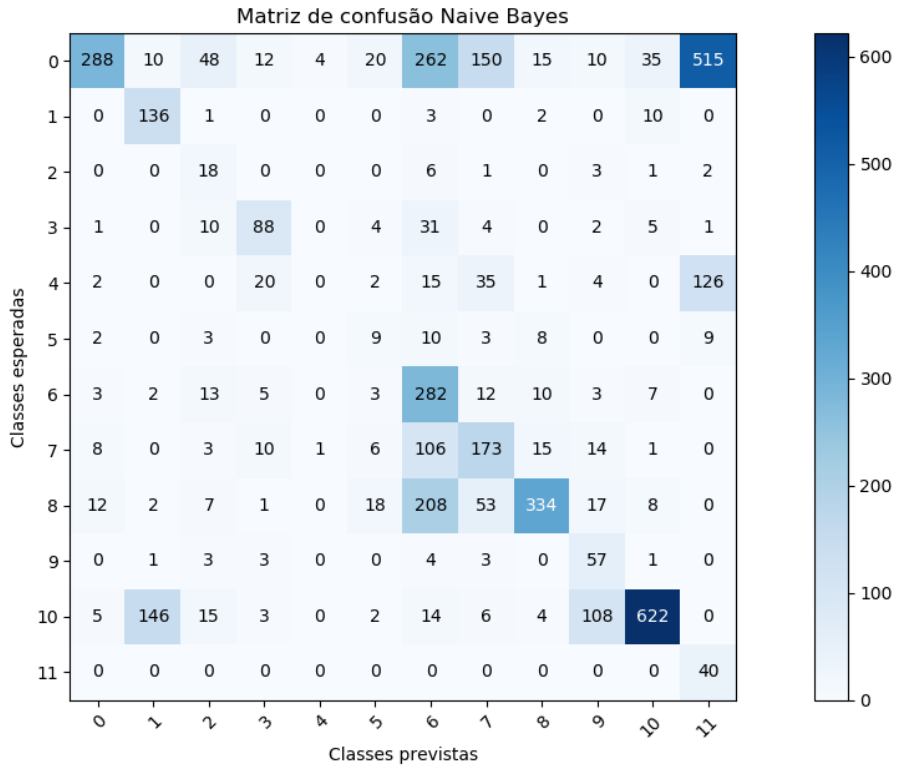

Figura 4.5 – Matriz de confusão do algoritmo *Naive Bayes* com *SMOTE*

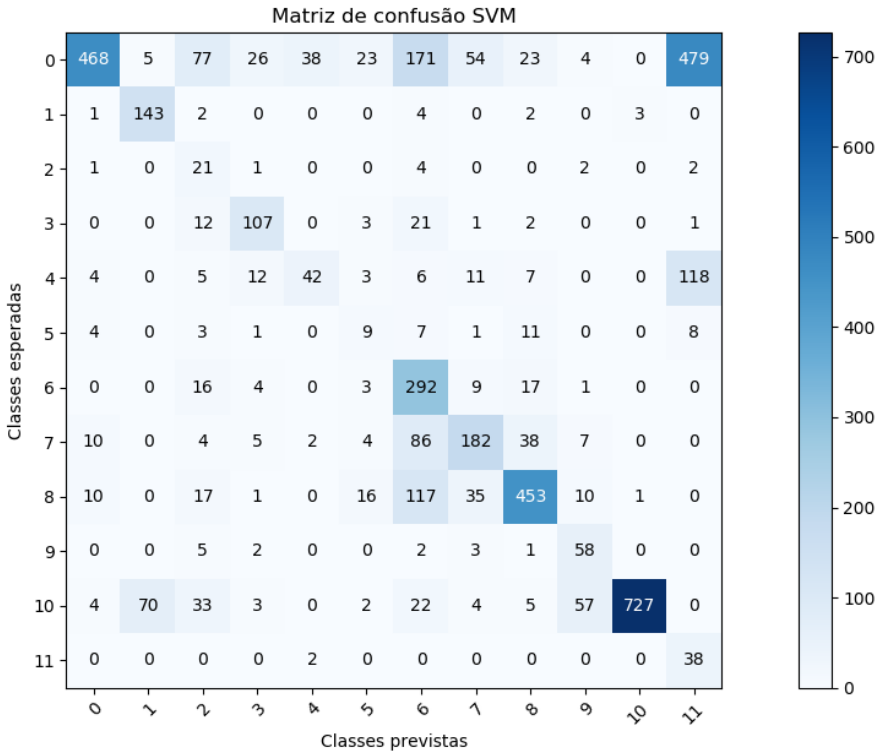

Figura 4.6 – Matriz de confusão do algoritmo *SVM* com *SMOTE*

O algoritmo *SVM* também teve bons resultados após a utilização do *SMOTE*, com destaque para os valores preditos da última classe,

4.2 Classificação de vários grupos de especialista

Da base com 165020 linhas de *logs*, 63935 incidentes foram resolvidos após serem encaminhados pelo *Service Desk* para dois ou mais grupos de especialistas. Com um total de 2829 caminhos diferentes, porém muitos deles muito pouco utilizados, sendo 1815 caminhos sendo utilizados apenas 1 vez por exemplo.

De todos esses caminhos foram selecionados os 10 mais frequentes para serem utilizados nos experimentos. Os caminhos são formados por 6 grupos de especialistas diferentes, com um tamanho mínimo de 2 e um máximo de 5. A distribuição de cada caminho pode ser visto na imagem 4.7.

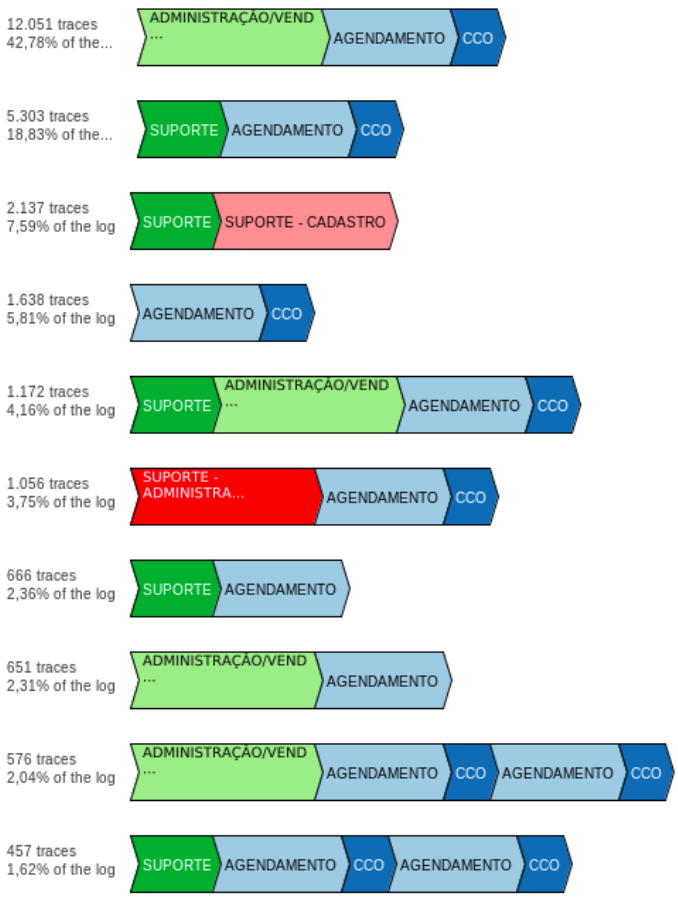

Figura 4.7 – 10 caminhos mais frequentes

4.2.1 Pré-processamento

Para representar a descrição dos *tickets* foi utilizado a mesma abordagem da sessão anterior 4.1, eliminando qualquer pontuação, números, datas e *stopwords*. Também foi aplicado a técnica de *stemming* e o dicionário ficou com um total de 13930 palavras diferentes. Toda execução também é feita utilizando *K-Fold Cross-Validation* com $k = 5$.

4.2.2 Resultados

	Precision	*Recall*	*F1*	*Accuracy*
$R - TRS_1$	0.454	0.397	0.388	0.693
$R - TRS_2$	0.574	0.408	0.335	0.256
$R - TRS_3$	0.400	0.394	0.310	0.385

Tabela 4.4 – Resultados grupo 2

Onde $R-TRS_1$ é a execução do algoritmo sobre a base, $R-TRS_2$ é a execução do algoritmo após aplicação da técnica *Cluster Centroids* e $R-TRS_3$ é a execução do algoritmo após a aplicação da técnica *Random Undersampling*.

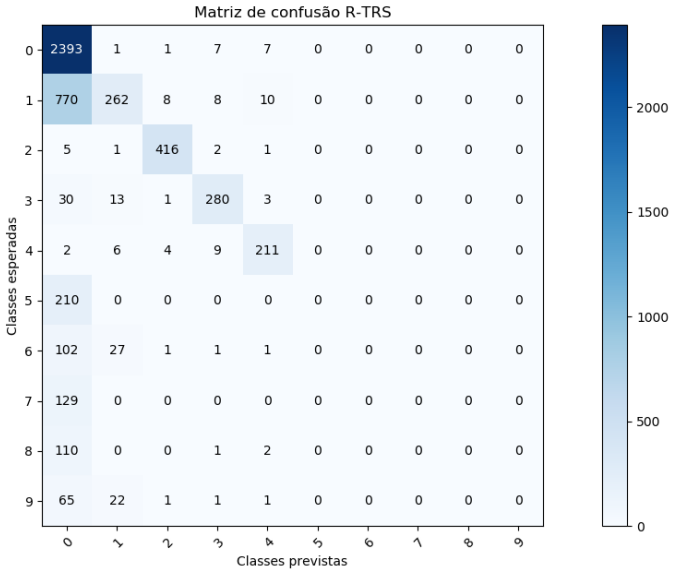

Figura 4.8 – Matriz de confusão do algoritmo *R-TRS*

Com um *F1-Score* de 0.388 e uma acurácia de 0.693 acertou bem os 5 caminhos mais bem representados na base, porém não teve um exito tão grande nos demais como pode ser visto na matriz de confusão 4.8. O problema foi amenizado com a utilização das técnicas de *Undersampling* que igualou o número de exemplos entre as classes na base.

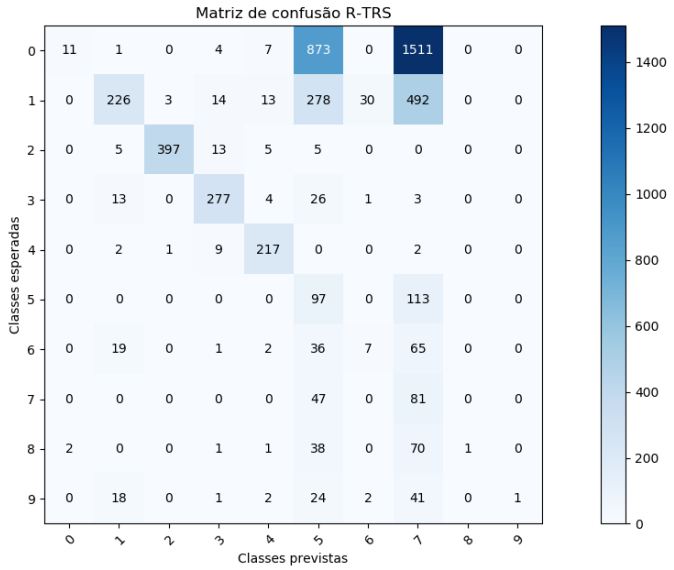

Figura 4.9 – Matriz de confusão do algoritmo *R-TRS* com *Cluster Centroids*

Com a aplicação da técnica *Cluster Centroids*, a escolha dos caminhos se tornou mais abrangente, espalhadas por todas as 10 classes definidas como pode ser visto na matriz de confusão 4.9 e a tabela 4.4 nos mostra que a precisão subiu para 0.574 porém a acurácia caiu mais da metade.

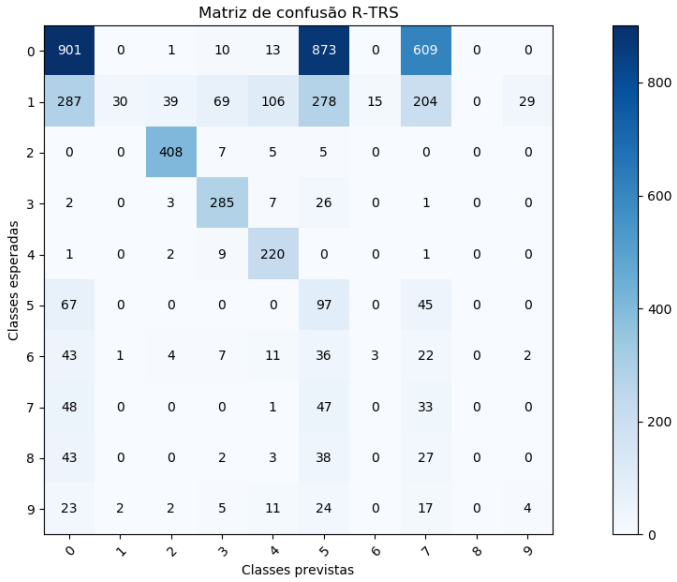

Figura 4.10 – Matriz de confusão do algoritmo *R-TRS* com *Random Undersampling*

Na matriz de confusão 4.10 da execução com *Random Undersampling* também mostrou bom espalhamento das classes preditas, porém novamente houve a queda das demais métricas ,

Capítulo 5

Conclusão

5.1 Conclusão

Foi apresentado um longo estudo em cima da base de incidentes catalogados, mostrado quais são os processos (caminhos) mais utilizados e os menos utilizados, também quais os grupos de especialistas mais acionados e os menos acionados de forma que esses dados possam auxiliar em tomadas de decisões futuras dentro da organização.

Utilizando técnicas de processamento de texto e classificadores foi apresentado um sistema que se mostrou apto a classificar automaticamente a qual grupo de especialistas um dado *ticket* deverá ser encaminhado. Com um *F1-Score* de 0.624 e uma acurácia de 0.744 o algoritmo *Knn* com $n = 2$ utilizando *SMOTE* teve o melhor resultado no conjunto de experimentos e se mostrou um bom classificador para as 12 classes utilizadas no primeiro grupo de experimentos.

O sistema também teve um resultado satisfatório na definição do caminho que um novo *ticket* deve seguir para ser resolvido. A melhor configuração apesar de ter uma *F1-Score* de 0.388 teve uma acurácia de 0.693 por acertar muito bem os 5 caminhos mais frequentes. Isso nos mostra que a seleção dos principais caminhos poderia ter sido menor ao invés dos 10 selecionados.

5.2 Trabalhos Futuros

Um ponto de melhoria pode ser visto nos experimentos com caminhos de múltiplos grupos de especialistas, o algoritmo apresentado teve um resultado satisfatório porém outras técnicas podem ser testadas como forma de comparação. Nesse mesmo experimento aplicar outras métricas de avaliação que não somente as clássicas de classificação para a avaliar a qualidade do encaminhamento.

No primeiro grupo de experimentos pode ser aplicado outros classificadores como Redes Neurais e técnicas de *ensamble* como Floresta Randômica ou *AdaBoost*.

Já no processamento de texto existem outras formas de representação que podem ser aplicadas, utilizar *Word Embeddings* por exemplo ou a variação do próprio *TF-IDF* utilizando uma composição variando os n gramas.

Referências Bibliográficas

Bouchaib Bahli and Suzanne Rivard. The information technology outsourcing risk: a transaction cost and agency theory-based perspective. *Journal of Information Technology*, 18(3):211–221, 2003.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer, 2009.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Su Mi Dahlgaard-Park. *The SAGE encyclopedia of quality and the service economy*. SAGE Publications, 2015.

T Dufresne and J Martin. Process modeling for e-business. *INFS 770 - Methods for Information Systems Engineering, Knowledge Management and E-Business Spring*, pages 1–28, 01 2003.

Alberto FernáNdez, Victoria LóPez, Mikel Galar, MaríA José Del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-based systems*, 42:97–110, 2013.

Matheus Correia Ferreira. Incident routing: Text classification, feature selection, imbalanced datasets, and concept drift in incident ticket management. Master's thesis, Universidade Federal do Rio de Janeiro, 2017.

Isabelle Guyon, B Boser, and Vladimir Vapnik. Automatic capacity tuning of very large vc-dimension classifiers. In *Advances in neural information processing systems*, pages 147–155, 1993.

Michael Hammer and James Champy. *Reengineering the Corporation: Manifesto for Business Revolution, A*. Zondervan, 2009.

AM Hay. The derivation of global estimates from a confusion matrix. *International Journal of Remote Sensing*, 9(8):1395–1398, 1988.

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

Jon Iden and Tom Roar Eikebrokk. Implementing it service management: A systematic literature review. *International Journal of Information Management*, 33(3):512–523, 2013.

ITIL. *The official introduction to the ITIL service lifecycle*. TSO,(The Stationary Office), 2007.

Mark W Johnson, Andrew Hately, Brent A Miller, and Robert Orr. Evolving standards for it service management. *IBM Systems Journal*, 46(3):583–597, 2007.

Shinji Kikuchi. Prediction of workloads in incident management based on incident ticket updating history. In *Proceedings of the 8th International Conference on Utility and Cloud Computing*, pages 333–340. IEEE Press, 2015.

Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. Supervised and traditional term weighting methods for automatic text categorization. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):721–735, 2008.

Spyros Makridakis. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529, 1993.

Ronny S Mans, Wil MP Van der Aalst, and Rob JB Vanwersch. *Process mining in healthcare: evaluating and exploiting operational healthcare processes*. Springer, 2015.

Mauricio Marrone, Francis Gacenga, Aileen Cater-Steel, Lutz Kolbe, et al. It service management: A cross-national study of itil adoption. *CAIS*, 34: 49, 2014.

João Menezes. Classificação de texto para melhoria no atendimento de help desk. Master's thesis, Universidade Federal do Rio de Janeiro, 2009.

Gengxin Miao, Louise E Moser, Xifeng Yan, Shu Tao, Yi Chen, and Nikos Anerousis. Generative models for ticket resolution in expert networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 733–742. ACM, 2010.

Gengxin Miao, Louise E Moser, Xifeng Yan, Shu Tao, Yi Chen, and Nikos Anerousis. Reliable ticket routing in expert networks. In *Reliable Knowledge Discovery*, pages 127–147. Springer, 2012.

Kayhan Moharreri, Jayashree Ramanathan, and Rajiv Ramnath. Probabilistic sequence modeling for trustworthy it servicing by collective expert networks. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, volume 1, pages 379–389. IEEE, 2016.

Sungbum Park and Young Sik Kang. A study of process mining-based business process innovation. *Procedia Computer Science*, 91:734–743, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

Thomas Roelleke. Information retrieval models: Foundations and relationships. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 5 (3):1–163, 2013.

Geary A Rummler and Alan P Brache. *Improving performance: How to manage the white space on the organization chart*. John Wiley & Sons, 2012.

I Sandvine. Global internet phenomena report. *North America and Latin America*, 2016.

Nauman Shahid, Nathanael Perraudin, Vassilis Kalofolias, Gilles Puy, and Pierre Vandergheynst. Fast robust pca on graphs. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):740–756, 2016.

Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

Aaron Smith and Monica Anderson. Online shopping and e-commerce. *Pew Research Center*, 19, 2016.

Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer, 2011.

Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter Van Den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *International Conference on Business Process Management*, pages 169–194. Springer, 2011.

Leo Van Selm. *ISO/CEI 20000-Introduction*. Van Haren, 1970.

Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

Geraldo Xexéo. Guia do orientado, 2018.

Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

Yan-Ping Zhang, Li-Na Zhang, and Yong-Cheng Wang. Cluster-based majority under-sampling approaches for class imbalance learning. In *2010 2nd IEEE International Conference on Information and Financial Engineering*, pages 400–404. IEEE, 2010.

Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Recommending ticket resolution using feature adaptation. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 15–21. IEEE, 2015.

Zhenhua Zhou. *The Development of Service Economy: A General Trend of the Changing Economy in China*. Springer, 2015.

Apêndice A

Resultados

Mais resultados dos experimentos realizados.

Knn	*Precision*	*Recall*	*F1*	*Accuracy*
Knn $n = 2$	0.676	0.628	0.638	0.793
Knn $n = 4$	0.721	0.635	0.660	0.810
Knn $n = 6$	0.717	0.632	0.656	0.812
Knn $n = 8$	0.728	0.626	0.655	0.817
Knn $n = 10$	0.729	0.619	0.650	0.817
Knn $n = 12$	0.715	0.616	0.642	0.796
Knn $n = 14$	0.720	0.610	0.641	0.806
Knn $n = 16$	0.707	0.610	0.637	0.797
Knn $n = 18$	0.712	0.608	0.637	0.796
Knn $n = 20$	0.707	0.607	0.635	0.795

Tabela A.1 – Resultados do algoritmo *Knn* com variação no número de vizinhos mais próximos n.

Knn	*Precision*	*Recall*	*F1*	*Accuracy*
Knn $n = 2$	0.617	0.672	0.624	0.744
Knn $n = 4$	0.592	0.681	0.607	0.718
Knn $n = 6$	0.574	0.692	0.598	0.694
Knn $n = 8$	0.564	0.690	0.588	0.688
Knn $n = 10$	0.564	0.693	0.588	0.684
Knn $n = 12$	0.561	0.692	0.585	0.681
Knn $n = 14$	0.555	0.691	0.577	0.677
Knn $n = 16$	0.565	0.703	0.576	0.670
Knn $n = 18$	0.572	0.714	0.577	0.663
Knn $n = 20$	0.576	0.714	0.576	0.660

Tabela A.2 – Resultados do algoritmo *Knn* com variação no número de vizinhos mais próximos n tendo aplicado o algoritmo *SMOTE*

SVM	*Precision*	*Recall*	*F1*	*Accuracy*
SVM rbf $C = 1$	0.540	0.366	0.374	0.707
SVM rbf $C = 2$	0.600	0.406	0.428	0.731
SVM rbf $C = 3$	0.588	0.435	0.459	0.744
SVM rbf $C = 4$	0.592	0.448	0.472	0.750
SVM rbf $C = 5$	0.596	0.459	0.484	0.755
SVM rbf $C = 6$	0.599	0.467	0.492	0.759
SVM rbf $C = 7$	0.598	0.474	0.498	0.763
SVM rbf $C = 8$	0.596	0.479	0.503	0.766
SVM rbf $C = 9$	0.595	0.483	0.507	0.768
SVM rbf $C = 10$	0.595	0.487	0.509	0.769
SVM linear $C = 1$	0.594	0.486	0.508	0.769
SVM linear $C = 2$	0.591	0.500	0.519	0.775
SVM linear $C = 3$	0.590	0.506	0.525	0.778
SVM linear $C = 4$	0.606	0.509	0.527	0.780
SVM linear $C = 5$	0.631	0.512	0.532	0.782
SVM linear $C = 6$	0.657	0.516	0.538	0.782
SVM linear $C = 7$	0.675	0.522	0.547	0.784
SVM linear $C = 8$	0.677	0.527	0.551	0.784
SVM linear $C = 9$	0.682	0.530	0.558	0.785
SVM linear $C = 10$	0.680	0.531	0.559	0.786
SVM poly $C = 1$	0.026	0.083	0.039	0.314
SVM poly $C = 2$	0.026	0.083	0.039	0.314
SVM poly $C = 3$	0.026	0.083	0.039	0.314
SVM poly $C = 4$	0.026	0.083	0.039	0.314
SVM poly $C = 5$	0.026	0.083	0.039	0.314
SVM poly $C = 6$	0.026	0.083	0.039	0.314
SVM poly $C = 7$	0.026	0.083	0.039	0.314
SVM poly $C = 8$	0.026	0.083	0.039	0.314
SVM poly $C = 9$	0.026	0.083	0.039	0.314
SVM poly $C = 10$	0.026	0.083	0.039	0.314

Tabela A.3 – Resultados do algoritmo *SVM* com variação nos parâmetros C e *kernel* utilizados.

SVM	*Precision*	*Recall*	*F1*	*Accuracy*
SVM rbf $C = 1$	0.539	0.629	0.492	0.580
SVM rbf $C = 2$	0.539	0.641	0.505	0.593
SVM rbf $C = 3$	0.541	0.648	0.513	0.599
SVM rbf $C = 4$	0.544	0.653	0.518	0.604
SVM rbf $C = 5$	0.545	0.654	0.520	0.605
SVM rbf $C = 6$	0.545	0.655	0.522	0.607
SVM rbf $C = 7$	0.547	0.658	0.524	0.609
SVM rbf $C = 8$	0.547	0.658	0.525	0.610
SVM rbf $C = 9$	0.548	0.659	0.526	0.611
SVM rbf $C = 10$	0.550	0.660	0.529	0.613
SVM linear $C = 1$	0.547	0.658	0.526	0.611
SVM linear $C = 2$	0.549	0.663	0.531	0.615
SVM linear $C = 3$	0.551	0.666	0.534	0.617
SVM linear $C = 4$	0.551	0.666	0.534	0.618
SVM linear $C = 5$	0.551	0.668	0.535	0.618
SVM linear $C = 6$	0.551	0.668	0.535	0.619
SVM linear $C = 7$	0.552	0.668	0.536	0.619
SVM linear $C = 8$	0.551	0.668	0.536	0.619
SVM linear $C = 9$	0.551	0.669	0.536	0.620
SVM linear $C = 10$	0.552	0.668	0.536	0.619
SVM poly $C = 1$	0.456	0.265	0.171	0.200
SVM poly $C = 2$	0.456	0.265	0.171	0.200
SVM poly $C = 3$	0.456	0.265	0.171	0.200
SVM poly $C = 4$	0.456	0.265	0.171	0.200
SVM poly $C = 5$	0.456	0.265	0.171	0.200
SVM poly $C = 6$	0.456	0.265	0.171	0.200
SVM poly $C = 7$	0.456	0.265	0.171	0.200
SVM poly $C = 8$	0.456	0.265	0.171	0.200
SVM poly $C = 9$	0.456	0.265	0.171	0.200
SVM poly $C = 10$	0.456	0.265	0.171	0.200

Tabela A.4 – Resultados do algoritmo *SVM* com variação nos parâmetros C e *kernel* utilizados tendo aplicado o algoritmo *SMOTE*