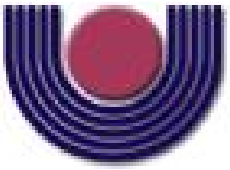

Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Análise de dados textuais para automação de um serviço de suporte online

Leonardo Aparecido Caracho

CASCAVEL
2018

Leonardo Aparecido Caracho

Análise de dados textuais para automação de um serviço de suporte online

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel.

Orientadora: Profª Drª Rosangela Villwock

CASCAVEL
2018

Leonardo Aparecido Caracho

Análise de dados textuais para automação de um serviço de suporte online

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Profª Drª Rosangela Villwock – orientadora
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Clodis Boscarioli
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. André Luiz Brun
Colegiado de Ciência da Computação,
UNIOESTE

Norldir Kunkel
Diretor de Inovação na Tecinco Tecnologia

Cascavel, 13 de dezembro de 2018

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

KDD *Knownledge Discovery in Databases*

KDT *Knownledge Discovery in Texts*

PLN Processamento de Linguagem Natural

VSM *Vector Space Model*

LSA *Latent Semantic Analysis*

SVD *Singular Value Decomposition*

Sumário

Resumo

Os avanços na área da tecnologia têm viabilizado a obtenção e armazenamento de grandes volumes de dados. Este trabalho teve como objetivo a extração de conhecimento a partir de uma coleção de documentos no formato textual, através da: aquisição, pré-processamento e agrupamento de dados. De posse dos resultados, criou-se um protótipo de *chatbot* para fazer a recuperação da informação e demonstrar o conhecimento adquirido. Para isso adotou-se diferentes estratégias de pré-processamento para a base de dados original e foram utilizadas diferentes abordagens, métodos e algoritmos para a realização do agrupamento. Com isso, buscou-se melhorar a qualidade dos grupos formados sendo estes avaliados por meio de um critério de avaliação interna. Dados os grupos criados foi possível desenvolver o protótipo do *chatbot*, o qual permitiu um fluxo de diálogo buscando a resolução de problemas.

Palavras-chave: KDT, KDD, *Machine Learning*, *Chatbot*, recuperação da informação.

Capítulo 1

Introdução

A quantidade de dados diariamente gerados e que não são utilizados para fins de obtenção de conhecimento é grande. É muito comum que as empresas, ao decorrer do tempo, produzam grandes quantidades de dados tornando-se um desafio extrair conhecimento útil a partir destes. Visando uma solução a este problema surgiu uma nova área denominada *KDD (Knowledge Discovery from Databases)*, que pode ser definida como o processo de extração de conhecimento útil a partir de bases de dados.

Mineração de dados é uma subárea deste processo, definida como a aplicação de algoritmos específicos para extrair padrões dos dados (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b). Mineração de dados é uma abordagem à análise de dados que engloba vários métodos baseados em técnicas de aprendizado de máquina, reconhecimento de padrões e estatística que engloba tarefas como: classificação, regressão, descoberta de regras de associação, agrupamento (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a). A tarefa a ser utilizada depende da base dados a ser analisada e do tipo de conhecimento que se deseja.

A base de dados a ser utilizada neste trabalho é formada por dados não estruturados, no formato texto, sendo imprescindível a realização de pré-processamento para eliminar palavras e textos sem valor e buscar representar esses dados em um formato que seja possível a aplicação de algoritmos de mineração de dados. Esta etapa é demorada, pois envolve muito tempo de processamento.

A extração de padrões de textos em português é um processo não trivial que envolve uma série de fatores, dado que a língua portuguesa apresenta diversas particularidades, como acentuação gráfica, semântica variada de acordo com o contexto de criação da sentença, além da variação linguística apresentada no território nacional. Assim, o desafio é extrair informação

útil a partir de coleções de documentos, as quais informações são identificadas em padrões presentes nos textos. (FELDMAN; SANGER et al., 2007).

A mineração de textos é utilizada para extrair conhecimento a partir de dados não estruturados. Esta etapa envolve o pré-processamento, que tem como objetivo remover informações inúteis dos textos, como assinaturas e *stopwords* para que, posteriormente, quando aplicados os algoritmos de mineração de dados, não induz a resultados imprecisos. A etapa de pré-processamento é também necessária para encontrar um modelo que possa ser utilizado para representar estes textos em um formato que os algoritmos possam reconhecê-los como parâmetros, uma vez que não existem algoritmos de mineração de dados adaptados para textos (REZENDE; MARCACINI; MOURA, 2011).

Os fins para que os dados são armazenados muitas vezes visa somente realizar consultas, não havendo a preocupação em como serão utilizados para obtenção de conhecimento. A mineração de dados é indispensável para que uma empresa entenda as necessidades e comportamento do seu consumidor, auxiliando na tomada de decisões e estratégias futuras de mercado. Se uma organização não utiliza alguma estratégia de gestão desses dados ela tende a ficar atrás de seus concorrentes, pois surgem novas necessidades de seus clientes que se forem percebidas a tempo, podem melhorar os índices de satisfação da empresa (ROBERTSON; WALKER, 1997).

Antes da aplicação de técnicas de extração de conhecimento, também é preciso entender o objetivo da empresa de posse da informação adquirida. Assim, pode-se implementar uma ferramenta capaz de realizar a gestão da informação gerada pelas etapas de mineração de dados, a partir de uma base de dados.

1.1 Motivação

A empresa na qual o trabalho foi realizado possui uma vasta quantidade de dados armazenados que não são utilizados para fins específicos, sendo um desperdício de potencial para estes dados. O presente trabalho teve como foco extrair conhecimento útil pra empresa a partir dados dados. Este processo envolve extração, pré-processamento, aplicação de algoritmos de mineração de dados e visualização dos resultados. O conhecimento gerado pode ser útil para melhorar o serviço de atendimento ao consumidor, tornando a interação entre suporte e cliente mais dinâmica.

1.2 Objetivos

O objetivo geral deste trabalho é extrair conhecimento a partir de uma base de dados textual e construir uma ferramenta para gestão da informação gerada pelo serviço de suporte de uma empresa da área de computação, podendo o resultado ser utilizado em qualquer empresa que tenha um sistema de gestão de atendimentos e/ou incidentes.

Como objetivos específicos citam-se:

- Implementar metodologia para aquisição e tratamento de dados não estruturados gerados pelo serviço de suporte de uma empresa;
- Desenvolver uma metodologia para rotular novos documentos;
- Definir uma metodologia que crie um fluxo de diálogo com um usuário, durante a interação com o suporte da empresa, de acordo com metodologia proposta no item anterior.

1.3 Organização

Além do Capítulo 1 introdutório, este documento está assim dividido:

- O Capítulo 2 provê uma contextualização da mineração de textos dentro do processo de descoberta de conhecimento.
- O Capítulo 3 traz uma visão do pré-processamento dos dados e as estratégias utilizadas.
- O Capítulo 4 discute os resultados obtidos.
- O Capítulo 5 apresenta as considerações finais e propostas para trabalhos futuros.

Capítulo 2

Processo de extração de conhecimento de dados não estruturados

A obtenção de conhecimento a partir de dados é uma evolução natural da tecnologia da informação. Os dados estruturados comumente armazenados em bancos de dados relacionais são os mais utilizados devido à uma maior facilidade de manuseio. No entanto, segundo (HAN; PEI; KAMBER, 2011), este processo pode utilizar dados de qualquer origem, como aqueles armazenados no formato textual. Isso demanda novos meios para o tratamento destes dados e para entender o que estas novas técnicas implicam no processo de *KDD* é preciso ter uma visão geral deste método.

2.1 *KDD* e Mineração de dados

Com o desenvolvimento humano, novas tecnologias surgem dia após dia. O que esse crescimento tem em comum em todas as áreas do conhecimento é o acúmulo de dados, os quais são uma rica fonte de informação e conhecimento quando aproveitados corretamente, o que faz com que seja necessário criar novas técnicas e ferramentas que auxiliem as pessoas na extração de conhecimento útil destes dados. Neste contexto, surge o *Knowledge discovery from databases (KDD)* (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b).

O Processo de *KDD* está em constante evolução e se relaciona a várias áreas do conhecimento como banco de dados, Aprendizado de Máquina, Reconhecimento de Padrões, Estatística, Inteligência Artificial, Aquisição de Conhecimento de Sistemas Especialistas, Visualização de Dados, Recuperação de Informação e Computação de Alto Desempenho. Aplicações de *KDD* se estendem por todos estes campos, com algoritmos e métodos e novas teorias (FAYYAD;

PIATETSKY-SHAPIRO; SMYTH, 1996a).

O *KDD* deve ser realizado através de uma série de etapas. As quais são específicas da aplicação dada ao processo exigindo atenção e intervenção do usuário em cada etapa, tornando este, um processo interativo e iterativo (PIATETSKY-SHAPIRO; FAYYAD; SMITH, 1996). A Figura 2.1 ilustra todas as etapas do processo de *KDD* onde os dados são coletados, pré-processados, passam pela etapa de mineração de dados, identificando padrões e por fim, são pós-processados para visualização do conhecimento obtido.

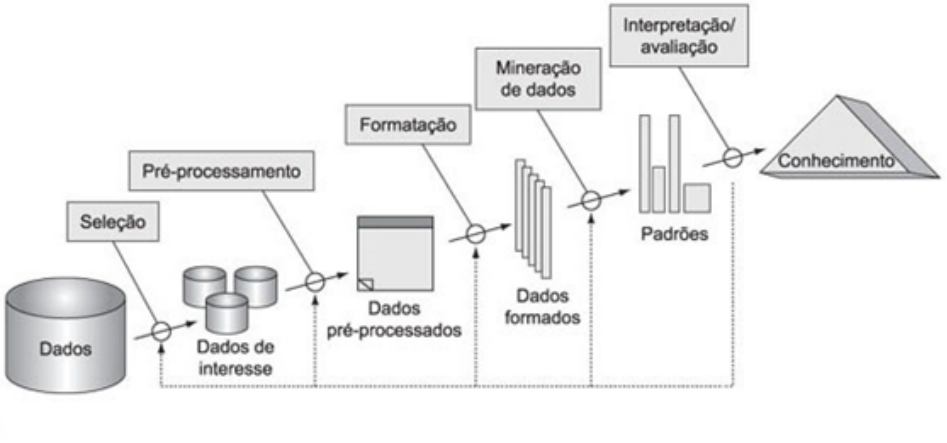

Figura 2.1: Etapas do processo de *KDD*
Fonte: (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b)

O processo de *KDD* é basicamente constituído de três partes principais: pré-processamento, mineração de dados e pós-processamento, sendo o objetivo principal transformar dados em fonte de conhecimento (GOEBEL; GRUENWALD, 1999).

A etapa de pré-processamento é iniciada após a aquisição dos dados e tem como finalidade tratar inconsistências nos dados, como dados discrepantes, dados faltantes e qualquer tipo de anomalia que possa interferir nos resultados de maneira negativa, invalidando assim o conhecimento obtido. Além de validar os dados, a etapa de pré-processamento deve também se certificar de que a maneira com que os dados estão armazenados será adequada aos algoritmos de mineração, que são aplicados na próxima etapa do processo (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a). Por fim, dá-se início à etapa de pós-processamento, que visa representar os resultados obtidos para que seja compreensível à interpretação humana.

A etapa de mineração de dados é onde os algoritmos e métodos de extração de padrões dos dados são aplicados. Esses métodos têm diferentes objetivos que dependem do resultado dese-

jado. Vários métodos diferentes podem ser aplicados sucessivamente para atingir o resultado desejado (GOEBEL; GRUENWALD, 1999).

2.2 Técnicas e tarefas de mineração de dados

É importante distinguir uma tarefa de uma técnica de mineração. A tarefa consiste na especificação do que queremos buscar nos dados, o tipo de regularidades ou categoria de padrões que temos interesse em encontrar, ou que tipo de padrões poderiam nos surpreender (por exemplo, um gasto exagerado de um cliente de cartão de crédito, fora dos padrões usuais de seus gastos). A técnica de mineração consiste na especificação de métodos que podem garantir como descobrir os padrões que nos interessam. Abaixo são listadas as principais tarefas de mineração de dados.

- **Regras de associação:** É um padrão da forma $X \rightarrow Y$, onde X e Y são conjuntos que podem expressar as compras feitas por um cliente, ou problemas relatados para o suporte. Para entender melhor este conceito, considere um serviço de suporte de uma empresa onde é relatado um problema X e também um problema Y, pode se dizer que ambos problemas têm relação entre si, facilitando sua resolução por parte do atendente.

- **Análise de padrões sequenciais**: São padrões que ocorrem frequentemente nos dados. Existem alguns tipos de padrões sequenciais como itens frequentes, sequencias frequentes e estruturas frequentes. Item frequente é, por exemplo, o item que é frequentemente comprado junto o produto cerveja, como carne, petiscos ou outro tipo relacionado. Padrão sequencial é dito quando se tem uma sequência de itens que são comprados em sucessão, como, por exemplo, uma pessoa em um restaurante pede um prato, depois um bebida e por fim uma sobremesa. Estruturas frequentes ocorrem quando se tem uma estrutura (grafos, árvores, etc) que se repete e a mineração destas estruturas pode levar a padrões interessantes nos dados.

- **Classificação:** Segundo (HAN; PEI; KAMBER, 2011), classificação é o processo de encontrar um modelo (função) que descreve e distingue classes de dados e conceitos. Sendo o modelo derivado e baseado na análise de um conjunto de dados de treino (dados

em que a classe que ele pertence é conhecida). O modelo é utilizado para predizer classes para objetos que a classe é desconhecida (teste).

- **Agrupamento:** Ao contrário da classificação, que rotula novos objetos a partir de objetos que tem classes rotuladas (conjunto de treino), o agrupamento rotula novos objetos sem consultar nenhum rótulo *a priori* (não supervisionado). Em muitos casos, dados rotulados não existem e o agrupamento pode ser utilizado para gerar rótulos para um conjunto de dados. Os grupos de dados são formados com base no principio de maximizar a similaridade intraclasse e minimizar a similaridade interclasse (HAN; PEI; KAMBER, 2011).

- **Avaliação de *outliers*:** São dados discrepantes que não seguem o padrão. São apresentados como anomalias nos dados, podendo ser entendidos como um ruído indesejável. Um exemplo de utilização seria em sistemas de detecção de fraude.

2.3 Mineração de textos

Considere, por exemplo, as mensagens enviadas via e-mail, onde tanto as mensagens recebidas quanto as enviadas não seguem um padrão e são apenas redigidas subjetivamente ao usuário, não havendo uma organização a priori. A questão é que essa fonte de dados contém conhecimento implícito relativo ao usuário que enviou aquela mensagem e esta informação pode caracterizar a mensagem de muitos modos, como a opinião do usuário sobre um produto, o sentimento do mesmo quanto a algum serviço, indicando um nível de satisfação e muitas outras informações relevantes. Basicamente, a mineração de textos é um conjunto de métodos usados para navegar, organizar, achar e descobrir informação em bases textuais (FELDMAN; SANGER et al., 2007).

Na Figura 2.2 é possível observar que existem dois ramos bem definidos na área de descoberta de conhecimento. O foco deste trabalho é a descoberta de conhecimento em textos *(Knowledge Discovery in Texts - KDT)*. A mineração de textos pode ser vista como uma extensão da etapa de mineração de dados, focada, entretanto, na análise de textos. Também é chamada de *Text Data Mining* ou *Knowledge Discovery in Texts* (ARANHA; PASSOS, 2006).

Os dados podem ser divididos em dois tipos: estruturados e não estruturados. Os primeiros

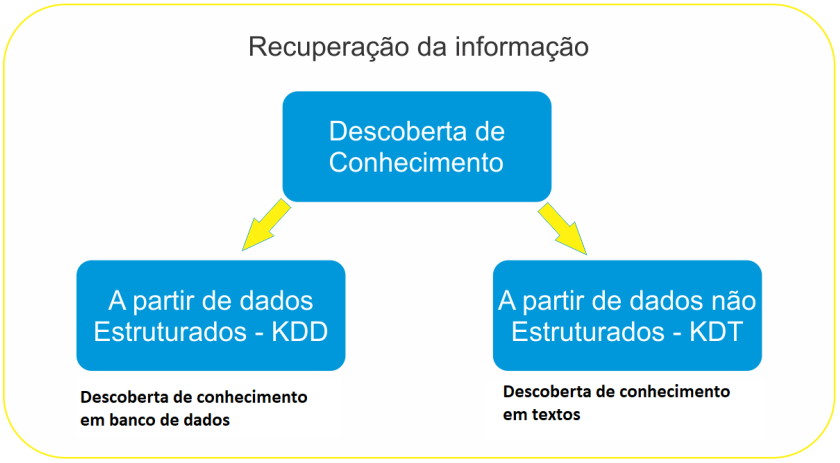

Figura 2.2: Processo de descoberta de conhecimento
Fonte: (MORAIS; AMBRÓSIO, 2007)

são aqueles armazenados em bancos de dados relacionais, divididos por linhas e colunas, de fácil recuperação. Dados não estruturados são aqueles de difícil recuperação por não serem organizados. É nesta categoria que se encontram os dados no formato textual.

A mineração de textos surgiu como alternativa para lidar com dados não estruturados e isso faz com que seja uma tarefa muito mais complexa que mineração de dados, uma vez que trabalha com dados textuais que são inerentemente bagunçados e desestruturados (TAN et al., 1999).

Como visto, textos fazem parte da categoria de dados não estruturados, isso quer dizer que antes de começar a trabalhar com eles é necessário estruturá-los, isto é, adotar algum procedimento que transforme a sequência de caracteres em objetos relacionados entre si. A lógica dessa transformação está presente no próprio texto, através de padrões linguísticos (ARANHA; PASSOS, 2006). É necessária a aplicação de técnicas de mineração de textos que estruturam estes dados para que o formato seja aceito por algoritmos de mineração de dados.

O processo de descoberta de conhecimento em textos consiste basicamente de duas fases: o tratamento e conversão do texto para um forma estruturada e em seguida a aplicação de algoritmos para extração de conhecimento (CORRÊA et al., 2003).

2.3.1 Processo de mineração em textos

Segundo (MORAIS; AMBRÓSIO, 2007) o processo de mineração de textos envolve uma série de etapas que são: Seleção de documentos, tipo de abordagem dos dados (Análise semân-

tica ou estatística), preparação dos dados, indexação e normalização, cálculo da relevância dos termos, seleção dos termos e pós-processamento. A Figura 2.3 ilustra o processo da forma que foi utilizada neste trabalho.

Figura 2.3: Processo de extração de conhecimento em textos
Fonte: Adaptado de (MORAIS; AMBRÓSIO, 2007)

Para iniciar o processo de descoberta de conhecimento nos textos é preciso definir o tipo de abordagem a ser utilizada. De acordo com (EBECKEN et al., 2003) há duas formas: análise semântica baseada na funcionalidade dos termos e a análise estatística baseada em frequência, descritas a seguir.

Análise Semântica

Avalia a sequência dos termos dentro do contexto da frase, para identificar corretamente a função da cada termo. A análise semântica utiliza técnicas de PLN (Processamento de linguagem natural) e é utilizada quando se quer uma qualidade melhor da mineração de textos em documentos com conteúdo semântico relevante.

Para processamento de linguagem natural é preciso ter, pelo menos, conhecimento morfológico, sintático, semântico e pragmático do discurso e do mundo (MORAIS; AMBRÓSIO, 2007).

- **Conhecimento morfológico** - Conhecimento da estrutura, da forma e das inflexões das

palavras.

- **Conhecimento sintático** - Conhecimento estrutural das listas de palavras, e como elas podem ser combinadas para produzir sentenças.
- **Conhecimento pragmático** - Conhecimento do uso da língua em contextos diferentes e como estes afetam seu significado e interpretação.
- **Conhecimento do discurso** - Como as sentenças precedentes afetam a interpretação da próxima sentença.
- **Conhecimento do mundo** - Conhecimento geral do domínio ou mundo com qual a linguagem natural se relaciona.

Técnicas de análise semântica procuram identificar a importância das palavras no texto, de acordo com o contexto em que elas se encontram. Para um único texto é possível definir um grau de relevância para os termos que aparecem dentro dele. Entretanto, para um processo maior como categorização, onde um documento é classificado por uma medida de similaridade, este processo é custoso e inviável.

Análise Estatística

Na análise estatística, segundo (MORAIS; AMBRÓSIO, 2007), a importância de um termo é relacionada com o número de vezes que ele aparece no texto. Seu processo envolve o aprendizado estatístico a partir de dados, onde estão incluídas as etapas de codificação e estimativa dos dados e modelos de representação de documentos.

- **Codificação dos dados** - Segundo (EBECKEN et al., 2003), uma codificação inicial dos dados é baseada nas indicações de um especialista ou algum critério que reflita as propriedades de interesse dos dados. Este passo contém as etapas de seleção de características, ou seja, nela é feita a eliminação de ruídos e remoção de termos indesejados. Importante lembrar que estes termos removidos não serão utilizados para a recuperação da informação. Isto torna esta etapa crítica para definição dos termos relevantes para a recuperação de informação.

- **Estimativa dos dados** - Esta etapa, segundo (MORAIS; AMBRÓSIO, 2007), envolve a procura de um modelo adequado a partir de um conjuntos de modelos (espaço de modelos). Um modelo pode ser obtido a partir da aplicação de um algoritmo de aprendizado ou de um método de estimativa.

- **Modelos de representação de documentos** - Um modelo que não leva em consideração a ordem das palavras mas a frequência que elas aparecem nos textos é conhecido como *bag-of-words*. Tal modelo transforma dados não estruturados em um formato estruturado, especificamente uma tabela atributo-valor (FELDMAN; SANGER et al., 2007). Esta técnica provê uma quantidade considerável de informações sobre associações entre palavras e documentos e têm se apresentado eficiente para o agrupamento e para recuperação de informações a partir de grandes coleções de textos (MORAIS; AMBRÓSIO, 2007).

Depois de definida a abordagem a ser utilizada, o passo seguinte é iniciar a etapa de preparação dos dados.

2.3.2 Preparação dos dados

Esta etapa envolve desde a aquisição dos dados à definição de um modelo de representação. Segundo (EBECKEN et al., 2003) esta fase engloba a seleção dos dados que constituirão a base de textos de interesse e o trabalho inicial para tentar solucionar o núcleo que melhor expressa o conteúdo dos textos, desprezando assim termos irrelevantes dos documentos.

O objetivo desta etapa é a redução da dimensionalidade e a identificação de similaridades nos textos. O pré-processamento visa extrair textos escritos em linguagem natural, inerentemente não estruturados, e transformá-los em uma representação estruturada, concisa e manipulável por algoritmos de agrupamento de textos. Para tal, são executadas atividades de tratamento e padronização na coleção de textos (FELDMAN; SANGER et al., 2007). Esta etapa envolve identificar e remover termos que não serão úteis para a análise, como assinaturas, palavras não significativas, redução à raiz da palavra, lematização, identificação das partes do discurso *(Speech tagging)*, onde o principal desafio é preservar as principais características do texto.

Na tokenização (identificação de termos), um documento é representado por uma coleção de termos, normalmente separados por espaços ou quebras de linha. A tokenização separa estes termos e os coloca em uma lista, onde cada termo é um elemento da lista. A tokenização também

pode ser realizada por sentenças, onde, ao invés de palavras, os documentos são separados por sentenças.

Uma pequena coleção de textos pode facilmente conter milhares de termos, muitos deles redundantes e desnecessários, que tornam lento o processo de extração de conhecimento e prejudicam a qualidade dos resultados (REZENDE; MARCACINI; MOURA, 2011). Sendo necessária a remoção destas palavras, definidas como *stopwords*. Primeiramente é criado uma lista contendo palavras compostas por artigos, pronomes, advérbios etc, dependendo do idioma dos documentos. Esta lista é denominada *stoplist*. Então o texto é dividido em um lista de termos, e é feita remoção do termo, caso ele pertença à *stoplist*.

O processo de *stemming* (Normalização morfológica) visa reduzir a palavra para o seu radical, reduzindo prefixos e sufixos, também com o intuito de reduzir as variações sintáticas das palavras. Em (EBECKEN et al., 2003) são descritos três métodos de *stemming*: método do *stemmer* S; método de Porter; método de Lovins.

O método do *stemmer* S, remove apenas sufixos de palavras, que geralmente formam o plural. O método de Porter, consiste na identificação de diferentes inflexões referentes à mesma palavra e sua substituição por um radical comum. Já o método de Lovins, remove cerca de 250 sufixos diferentes para palavras da língua inglesa. Seu algoritmo remove apenas um sufixo por palavra, retirando o sufixo mais longo conectado à mesma (MORAIS; AMBRÓSIO, 2007).

Lematizar uma palavra (Lematização) tem como objetivo, assim como o *stemmer*, reduzir as variações sintáticas das palavras. Nessa abordagem, diferentemente do *stemming*, o foco não é reduzir os sufixos e prefixos de uma palavra, e sim usar os verbos no infinitivo, não afetando tanto a estrutura da palavra.

Segundo (YE et al., 2016) *Part-of-speech (POS) tagging* é o processo para classificar em qual categoria léxica uma palavra se encontra, como classificar uma palavra em advérbio, pronome, etc, podendo assim realizar a remoção de termos pela categoria que ele se encontra. Por exemplo, fazer remoção de pronomes presentes no texto.

As palavras mais frequentes em um texto costumam ter um significado importante, com exceção das *stopwords* que são palavras vazias e se repetem inúmeras vezes ao longo do texto. Identificar essas palavras mais significativas é um passo crucial para a extração de conhecimento de uma base de dados não estruturados. Para isso, é preciso identificar o peso de cada palavra.

Nesse contexto, o peso indica o grau de relação entre a palavra e os documentos em que ela aparece (WIVES, 2002). As fórmulas mais utilizadas para se calcular o peso de uma palavra são: frequência absoluta, frequência relativa, frequência inversa de documentos.

Frequência absoluta - Mede o número de vezes que um termo aparece no documento. É a medida mais simples de se estimar o peso de um palavra. No entanto, não é capaz de distinguir termos que aparecem em poucos documentos e termos que aparecem em muitos documentos.

Frequência relativa - Mede o número de vezes que um termo aparece no documento. Ao contrário da frequência absoluta, ele verifica quantos termos possui o documento, normalizando os pesos. Esta normalização faz com que os documentos formados por mais termos sejam escalados diferentemente de documentos compostos de poucos termos. A frequência relativa de um termo *x* ($F_{rel}x$) é calculada dividindo sua frequência absoluta (F_{abs}) pelo número total de termos do documento (*N*).

$$F_{rel}x = \frac{F_{abs}x}{N} \tag{2.1}$$

Frequência inversa de documentos - A frequência inversa de um termo *x* é a frequência absoluta dividida pelo número de documentos que ele aparece. Segundo (ROBERTSON; WALKER, 1997) a frequência inversa é capaz de aumentar a importância de termos que aparecem em poucos documentos e diminuir a importância de termos que aparecem em muitos documentos.

$$Peso_t = \frac{Freq_t}{DocFreq_t} \tag{2.2}$$

A fórmula acima, segundo (SALTON; BUCKLEY, 1988), mostra que o peso do termo é equivalente à relação entre o termo *t* e o documento *d*, sendo $Freq_t$ o número de vezes que o termo *t* aparece no documento *d*. $DocFreq_t$ é o números de documentos *d* que possui o termo *t*. No entanto, (WIVES, 2002) diz que coleção pode variar devido à adição de novos documentos ou devido às mudanças no conteúdo dos documentos (que podem ser modificados). Nesse caso, torna-se necessário recalcular periodicamente os pesos das palavras ou adotar alguma outra solução.

2.3.3 Recuperação de Informação

Um documento textual é um modelo de difícil abstração para algoritmos de *machine learning*. Para tanto, é utilizado um sistema de recuperação de informações. Um sistema de recuperação textual é desenvolvido para indexar e recuperar documentos do tipo textual, ou seja, documentos cujas informações estão descritas através da linguagem natural. Na literatura existe um ampla taxonomia de modelos, entre eles o booleano, probabilístico, fuzzy, busca direta etc. O método usado neste trabalho é o *Vector Space Model (VSM)* por ser baseado na frequência das palavras (ARANHA; PASSOS, 2006), favorecendo uma abordagem estatística, além de ser um dos modelos mais utilizados em sistemas de recuperação da informação (REZENDE; MARCACINI; MOURA, 2011).

O *VSM* representa documentos de uma coleção como vetores em um espaço multidimensional (Figura 2.4), onde cada eixo *Termo* da imagem representa a frequência de uma palavra. O vetor *Obj* representa um documento da coleção. Também é possível notar que quanto menor o ângulo formado entre dois vetores, mais similares são os documentos e quanto maior o ângulo formado menor é a semelhança entre eles. Na Tabela 2.1 a cada linha apresenta-se a representação de um documento sendo os valores fornecidos pelo modelo *f*(t_i, d_j) na qual t_i é a frequência da palavra *i* no documento *j* dividido pelo total de palavras no documento *j*.

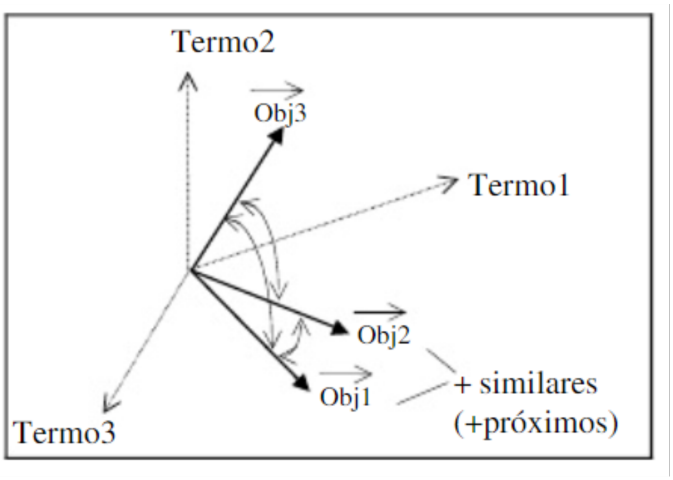

Figura 2.4: *Vector Space Model - VSM*
Fonte: (WIVES, 2002)

$Documento(Objeto)$	$Termo_{p1}$	$Termo_{p2}$	...	$Termo_{pi}$
D_1	f(p_1, d_1)	f(p_2, d_1)	...	f(p_i, d_1)
D_2	f(p_1, d_2)	f(p_2, d_2)	...	f(p_i, d_2)
...	...	...	...	...
D_j	f(p_1, d_j)	f(p_2, d_j)	...	f(p_i, d_j)

Tabela 2.1: Representação de documentos modelo *VSM*

Algumas desvantagens deste modelo dão-se com relação à sensibilidade semântica, onde documentos com contextos semelhantes mas com vocabulário diferente não serão associados. Assim como a ordem que os termos aparecem é perdida na representação vetorial, a informação semântica e sintática também é perdida, já que o *VSM* se baseia na independência dos termos de um documento. Definida a representação a ser utilizada, pode-se iniciar o processo de extração de padrões nos dados, que consiste na análise e interpretação de resultados gerados por algoritmos de mineração de dados. As tarefas mais usadas são classificação e agrupamento, sendo que o foco neste trabalho será no uso de agrupamento.

2.4 Agrupamento de textos

Na tarefa de agrupamento o objetivo é organizar um conjunto de objetos em grupos com base em uma medida de semelhança, sendo que os objetos pertencentes a um grupo são altamente similares entre si, mas dissimilares em relação a objetos de outros grupos (HAN; PEI; KAMBER, 2011). O objetivo da aplicação de agrupamento é maximizar a similaridade de documentos de um grupo (intragrupo) e minimizar a similaridade entre os grupos (intergrupos). Segundo (REZENDE; MARCACINI; MOURA, 2011) o processo de agrupamento depende de dois fatores principais: uma medida de proximidade e uma estratégia de agrupamento.

2.4.1 Medidas de proximidade

Uma medida de proximidade é fundamental para um algoritmo de agrupamento por medir a similaridade entre documentos e, posteriormente, definir para qual grupo eles pertencem. As medidas de similaridade mais utilizadas em dados textuais são as medidas *Cosseno* e *Jaccard* (REZENDE; MARCACINI; MOURA, 2011).

Considere dois documentos representados por x = (x_1, x_2, ...,x_n) e y = (y_1, y_2, ... , y_n) sendo que cada elemento de x e y representa a frequência de uma palavra de seu respectivo

documento, eles estão representados pelo vetor n dimensional onde cada uma das palavras é equivalente a uma dimensão do vetor. A medida cosseno é definida como o cosseno do ângulo formado entre os vetores desses dois documentos, conforme a Equação 2.3 abaixo (FELDMAN; SANGER et al., 2007).

$$cosseno(x, y) = \frac{x \bullet y}{|x||y|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2} \tag{2.3}$$

O resultado da equação acima varia de 0 a 1, sendo que se o resultado for 0, o ângulo formado entre os vetores x_i e x_j é de 90, não compartilhando nenhum termo em comum. Caso seja 1, o ângulo formado entre x_i e x_j é 0, significando que os dois documentos são iguais.

A medida de proximidade de Jaccard é mais indicada caso os vetores sejam representados por valores binários, indicando a presença ou ausência de algum termo (REZENDE; MARCACINI; MOURA, 2011). Seja x_i e x_j dois documentos, a proximidade entre eles por Jaccard é dada segundo a Equação 2.4:

$$jaccard(x_i, x_j) = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \tag{2.4}$$

Onde f_{11} é o número de palavras presentes nos dois documentos, f_{01} o número de palavras ausentes em x_i e presentes em x_j e f_{10} o número de palavras presentes em x_i e ausentes em x_j. O valor resultante da fórmula acima fica no intervalo [0,1], sendo que, quanto mais próximo de 1, maior é a similaridade.

2.4.2 Métodos de agrupamento

Segundo (FELDMAN; SANGER et al., 2007) agrupamento é um processo não supervisionado, não qual objetos são classificados em grupos (*clusters*). Estes grupos são criados com base em características extraídas do próprio conjunto de dados. Um exemplo de agrupamento seria a separação de homem e mulher, tendo como base características como peso, altura e formato do rosto. Seria possível criar dois grupos com características distintas, dizendo com base nestes atributos, o sexo de uma pessoa, isso sem consultar nenhum rótulo *a priori* e somente os próprios dados.

Segundo (FELDMAN; SANGER et al., 2007) a maioria das técnicas de agrupamento foram desenvolvidas para atuar em dados estruturados. Somente com o crescimento da internet a

extração de conhecimento baseado em textos ganhou importância.

O enfoque no agrupamento por documentos pode ser justificado por que um conjunto de grupos de documentos similares pode ser armazenado em seções. Assim, quando uma consulta por um documento for feita, a resposta à consulta não será apenas um documento e sim o próprio grupo que contém o documento, que podem ser ou não relevantes à mesma consulta.

Esta afirmação parte do pressuposto de que todos os documentos de um mesmo grupo tratam de um assunto semelhante e que grupos diferentes possuem documentos que tratam de assuntos divergentes. Esta técnica tem o intuito de agrupar os documentos por assuntos similares, tornando mais fácil a manipulação e visualização da informação.

Segundo (CUTTING et al., 1992) há dois tipos de agrupamento: agrupamento por partição e agrupamento hierárquico, ambos se referem à forma de construção dos grupos. No Agrupamento por partição, os documentos são agrupados em n grupos distintos, onde o valor de n é predeterminado. O grau de semelhança de um grupo é definido com base nas características em comum dos elementos que o compõe. A principal desvantagem, segundo (WIVES, 2004), é não haver uma estrutura de relações que indique os assuntos mais abrangentes, os assuntos mais específicos e os inter-relacionados entre eles, tornando difícil ao usuário encontrar a informação que necessita.

Seja um conjunto de dados D em um agrupamento hierárquico o conjunto D é decomposto hierarquicamente e representado por um dendograma, que é uma arvore que divide D em subconjuntos menores, até que cada subconjunto seja representado por um só elemento, nesta hierarquia cada nó da árvore representa um agrupamento de D o dendograma pode ser criado das folhas a raiz *bottom-up*(abordagem aglomerativa) ou a partir da raiz para as folhas *top-down* (abordagem divisiva).

Os grupos identificados são analisados recursivamente, identificando as relações entre grupos, ao contrário dos particionados onde os algoritmos realizam somente um passo de processamento. Os hierárquicos podem ser de dois tipos: aglomerativo e global, em ambos os casos é produzida uma árvore onde as folhas representam os elementos individuais e os nós intermediários correspondem aos grupos formados pelo agrupamento de seus próprios filhos.

Além destes dois tipos de agrupamentos, há também o agrupamento baseado em densidade. A ideia deste método é que para cada ponto do conjunto de dados há um raio de vizinhança

EPS, para cada $EPS > 0$ deve conter um número mínimo de pontos para ser considerada um *cluster* (SANDER et al., 1998). A ideia consiste em localizar ou determinar regiões de alta densidade que estejam separadas por regiões de baixa densidade.

Os algoritmos mais utilizados em *KDT* são *Star*, *K-means* e hierárquico. Deste três algoritmos, o *Star* não apresentou bons resultados nos primeiros testes portanto foi optado por utilizar o *K-means*, em conjunto com o *DBSCAN*, buscando resultados mais eficazes.

2.4.3 Algoritmo *Star*

O algoritmo de agrupamento *Star* aplica grafos de similaridade ponderados e não direcionados. Considere V o conjunto de vértices, E o conjunto de arestas, w o peso e uma aresta ponderada E_p, sendo G um grafo tal que $G = (V, E, w)$ forma-se um subgrafo $G' = (V, E_p)$ baseado em G para organizar a informação. O algoritmo *Star* possui muitas variantes sendo que a utilizada neste trabalho é o K-Star (SHIN; HAN, 2003). As vantagem do *K-Star* é que possibilita a detecção do número de grupos formados de modo automático em contraste com outros algoritmos de agrupamentos por partição como o *K-means*. Esta abordagem forma grupos mais compactos e não são afetados pela presença de *outliers*. Entretanto, uma desvantagem é a formação de grupos independentes que poderiam resultar em um único grupo.

O ponto de corte T (*thresold*) representa uma distância, que define se um elemento pertencerá ou não ao grupo, o valor de T é definido como uma média dos valores da matriz de similaridade, que é responsável por armazenar a distância ou similaridade entre os documentos de uma coleção, o algoritmo é descrito por (PINTO et al., 2009) através das seguintes etapas.

1. O primeiro passo consiste em procurar o valor máximo na matriz de similaridade $G(d_i,d_j)$ e formar um grupo (C_i) composto pelos documentos i e j. Estes documentos são marcados para não serem utilizados novamente.

2. Para cada documento não marcado (d_k) na linha i da matriz, se $G(d_k,d_i) > T$, então (d_k) é adicionado ao grupo (C_i) e (d_k) é marcado.

3. Retorna para o passo 1

2.4.4 Algoritmos *K-means* e *MiniBatchKmeans*

Dentre os métodos de agrupamento particionados o *K-means* é o mais conhecido, e muito utilizado para coleções de textos (STEINBACH et al., 2000). Como visto anteriormente, os textos podem ser representados como vetores em um espaço n-dimensional. Dado um conjunto de vetores (x_1, x_2, ...,x_n) o algoritmo particiona estes vetores em um conjunto de k grupos, sendo k um número pré-definido de grupos. Cada grupo tem um respectivo centróide que é um vetor médio computado a partir dos vetores do grupo. A equação 2.5 abaixo define o centróide C como sendo igual a soma dos n elementos pertencentes a um grupo G dividido pelo número de elementos presentes no grupo. Mantendo assim características centrais do grupo e criando um elemento que representa o grupo preservando as suas características mais gerais.

$$C = \frac{1}{|G|} \sum_{x \in G} x \tag{2.5}$$

O critério de parada do algoritmo pode ser até o algoritmo convergir a um estado que o centróide não sofre mais alterações, ou também quando atingir um número x de iterações. O algoritmo *k-means* é popular por sua simplicidade e eficiência. No entanto, a seleção das sementes do grupo pode ser um problema se não for uma boa escolha, gerando grupos sub-ótimos. Além disso, o número pré-definido de grupos pode não ser uma boa escolha para o problema abordado. Abaixo é mostrada os passos do algoritmo para um conjunto de documentos X = $(x_1, x_2, ..., x_n)$ com k grupos.

Algoritmo *K-means*:

1. Seleciona-se aleatoriamente k documentos como centróides iniciais.
2. Para cada documento x pertencente a X se computa a similaridade de x para cada centróide C e atribui x ao centróide mais próximo.
3. Recalcula-se o centróide de cada grupo.
4. Repete passo 2 até atingir o critério de parada.

O algoritmo *MiniBatchKmeans* é uma variação do *K-means* no qual usa *mini-batches* para reduzir o tempo de computação. *Mini-batches* são subconjuntos gerados aleatoriamente a partir

dos dados que estão sendo analizados, estes subconjuntos tem o intuito de reduzir o processamento necessário para o algoritmo convergir. O algoritmo interage sobre os passos, descritos a seguir.

Algoritmo *MiniBatchKmeans*:

1. Seleciona-se aleatoriamente k documentos como centróides iniciais.
2. No primeiro passo um número aleatório de elementos próximos formam um *mini-batch*, que são designados ao centróide mais próximo.
3. No segundo passo, os centróides são atualizados, ao contrário do *K-means* cada centroide é calculado pela média dos elementos de cada *mini-batch* que a ele pertence.
4. Repete passo 2 até convergir ou atingir um critério de parada.

2.4.5 Algoritmo *DBSCAN*

O algoritmo DBSCAN (*Density Based Spatial Clustering of Applications with Noise*) é um algoritmo de agrupamento por densidade, cujo o objetivo é agrupar regiões de alta densidade separadas por regiões de baixa densidade (ESTER et al., 1996). Existem dois parâmetros que se deve definir para executar o algoritmo, que são: *Eps* e *MinPts*. O *Eps* é um raio que avalia para um determinado ponto o número de pontos ao seu redor. Já o *MinPts* define se, para o ponto em questão, o número de pontos ao seu redor é maior do que o *MinPts* especificado. Caso seja, este ponto é definido como um ponto central. Pontos de limite são aqueles que se encontram dentro da vizinhança de um ponto central mas que não tem o número mínimo de vizinhos. Caso o ponto não seja nem central e nem de limite ele é caracterizado como um ponto de ruído.

Algoritmo DBSCAN:

1. Inicialmente se classifica todos os pontos como sendo ruído, limite ou central.
2. Os pontos de ruídos são eliminados.
3. Colocar arestas entre todos os pares de pontos de centro que estejam um dentro do raio do outro.
4. Tornar cada grupo de pontos de centro e seus pontos de limite conectados formando um grupo separado.

2.5 Validação do agrupamento

A etapa final consiste em avaliar se o conhecimento adquirido é válido. Esta análise pode ser feita de modo subjetivo com o auxílio de um especialista ou por meio de índices estatísticos. Esta etapa se baseia em uma métrica que quantifica alguma informação sobre a qualidade do agrupamento. Segundo (EVERITT; LANDAU; LEESE, 2001) existem três tipos de critérios para validação de agrupamento: critérios internos, relativos e externos.

Os critérios internos medem a qualidade de um agrupamento a partir de informações do próprio conjunto de dados. Um critério interno analisa se as posições dos objetos do agrupamento obtido são correspondentes à matriz de similaridade. Os critérios relativos comparam diversos agrupamentos para decidir qual deles é mais adequado para os dados.

Os critérios externos correspondem a avaliar o agrupamento baseado em uma informação externa, que pode ser uma informação que o próprio analisador tem da estrutura dos dados ou por um especialista do domínio, sendo necessário que se tenha um conhecimento a priori do conjunto de dados, conhecendo como um conjunto ideal deveria ser e comparando-o com o obtido.

Na validação baseada em critérios externos a estrutura dos dados é baseada em uma estrutura pré determinada, denominada estrutura de teste (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2002a). Esta estrutura de teste denominada ideal é construída com base no conhecimento a priori definido por um especialista.

Na validação baseada em critérios relativos, segundo (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2002b), os grupos são avaliados e validados comparando-os com a estrutura de grupos resultantes com outras estruturas ou esquemas gerados pelo mesmo algoritmo, executando diferentes parâmetros de entrada. Nesse caso, o mesmo algoritmo é executado repetidamente, testando todos os valores possíveis para cada um de seus parâmetros, identificando assim o melhor parâmetro para o conjunto de dados em questão.

Existe uma variedade de métodos de agrupamento para os mais variados tipos de dados. Saber qual é método certo para um determinado tipo de dado é um desafio, uma vez que eles são desenvolvidos para tipos específicos de dados. Por isso, como dito em (WIVES, 2004), o usuário deve tomar o cuidado de escolher o método de agrupamento mais adequado ao tipo de dado que está manipulando.

A validação por critérios internos foi utilizada neste trabalho, por ser um método de avaliação não supervisionado, baseando a qualidade dos grupos formados somente por características dos próprios dados. No próximo capítulo é descrito o processo de aquisição e de preparação dos dados para a etapa de agrupamento assim como o critério de avaliação interna utilizado.

Capítulo 3

Aquisição e preparação dos dados

Na etapa inicial do trabalho foi preciso ter conhecimento do problema da empresa. Nesta etapa o foco se deu nos dados que foram utilizados e a forma como estavam estruturados. Esse contato inicial definiu que tipo de abordagem seria utilizada. O sistema utilizado dentro da empresa é o *OTRS (Open-source Ticket Request System)*, responsável por gerenciar o fluxo de mensagens entre cliente e o serviço de suporte da empresa. Nele que os chamados são recebidos quando o usuário relata algum problema ou reclamação, referente ao software proprietário por ele utilizado e produto da empresa. Este capítulo define algumas estratégias para tratar a base de dados e prepará-los para a aplicação dos algoritmos de mineração de dados.

3.1 *Open Ticket Request System*

O *OTRS* ou sistema livre de requisição de chamados é um sistema de gerenciamento de incidentes livre e de código aberto que opera no ambiente web. É usualmente utilizado em empresas e organizações para atribuir rótulos à requisições e rastreá-las mantendo-se assim um acompanhamento para futuras comunicações. Comumente, este serviço é utilizado para gerenciar reclamações, pedidos de suporte, relatórios de defeito e outras comunicações.

Dentro do *OTRS*, cada rótulo gerado é tratado de forma que todas as comunicações sobre ele se mantenham registradas, formando-se um "arquivo" que mostra o que aconteceu com o rótulo durante o tempo que ele esteve aberto. O *OTRS* permite que múltiplos agentes possam trabalhar simultaneamente em um mesmo chamado, ler mensagens e respondê-las de forma ordenada. Uma das vantagens de se implementar um sistema *OTRS* dá-se pela sua alta escalabilidade, permitindo se trabalhar com milhares de rótulos por dia, com um alto número de agentes

trabalhando simultaneamente.

Na empresa alvo desse estudo, o suporte é dividido em níveis organizados visando atender problemas em ordem crescente de complexidade, em que problemas mais simples são resolvidos no primeiro nível (suporte) e problemas mais complexos são destinados aos níveis posteriores (desenvolvimento). Quando um problema é relatado via *OTRS*, é atribuído a ele um id de identificação ou rótulo, sendo assim rotulado como um *ticket* único no sistema. O *ticket* é analisado pelo primeiro nível do suporte e pode ser repassado aos níveis seguintes dependendo de sua resolução. A interação com o cliente por meio deste serviço registra, a partir do *ticket* de abertura, todas as informações de mensagens trocadas posteriormente.

O *OTRS* é utilizado dentro da empresa para gerenciar a entrada de requisições e acompanhar comunicações futuras sobre elas: quando um usuário abre um chamado, ele é atribuído a uma fila de prioridade que fica aguardando a sua abertura pelo primeiro nível do suporte e assim é feita a análise do problema, podendo este ser repassado ou não aos níveis seguintes. Caso o chamado seja resolvido, ele é dado como finalizado e as interações acerca do mesmo são armazenadas no banco de dados da empresa.

3.2 Estrutura e armazenamento dos dados

O sistema gerenciador de banco de dados relacional (SGBDR) empregado na empresa é o MySQL e o software SQLyog foi utilizado para visualização e extração dos dados.

Antes de realizar o processo de extração da base de dados é preciso definir as tabelas e colunas que serão extraídas, dada a quantidade de dados armazenadas sobre cada *ticket*, sendo preciso separar aqueles que serão utilizados na descrição do problema daqueles que não serão aproveitados. Com isso pode se excluir as colunas de informações que não serão relevantes. Por exemplo, data e horário da mensagem, endereço de e-mail do remetente, entre outros. Restando apenas o id equivalente ao rótulo do *ticket* e o corpo das mensagens trocadas entre usuário e suporte.

Após a separação das colunas que serão exportadas, ainda existe uma enorme quantidade de dados que não agregaram à análise. Estes dados estão armazenados no formato de interações dentro do corpo do texto. Estas interações representam a ordem em que as mensagens chegam quando um *ticket* é aberto e rotulado. Quando um cliente abre um chamado e envia uma

mensagem ao suporte, esta mensagem vai ser a primeira interação, depois disso é enviada uma mensagem automática do próprio sistema confirmando a abertura do *ticket*. Enquanto isso, um atendente do suporte abre e analisa o chamado e faz um retorno ao cliente. Mensagens entre cliente e atendente são trocadas até o fechamento do chamado. Ou seja, até a resolução do problema. Uma abertura pode resultar em múltiplas iterações entre cliente e atendente.

As várias iterações entre cliente e atendente resultam em muita informação. Devido a este grande número de textos, se tornaria inviável processar todo o conteúdo. Neste contexto, a informação a ser utilizada será somente relativa à primeira iteração do cliente, ou à mensagem de abertura de chamado, dado que é nela que o usuário descreve o problema que será posto em primeira análise pelo primeiro nível do suporte.

3.3 Processo de extração dos dados

Uma vez definidos os dados de interesse, iniciou o processo de extração. Os dados foram exportados no formato *JSON (Javascript Object Notation)* devido à sua simplicidade e por ser um formato de fácil leitura, e com uma ampla variedade de bibliotecas para se trabalhar. Além disso, seu tamanho reduzido também proporciona uma maior velocidade na execução e transporte dos dados.

Um *script SQL* foi criado para exportar apenas as primeiras interações de todos os *tickets* armazenados no banco de dados, por meio do SQLyog, possibilitando a extração dos textos no formato *JSON*. A Figura 3.1 mostra o formato obtido após a extração da interação inicial feita pelo cliente, onde o valor 1 é referente ao id único da mensagem, 2 o assunto e 3 à mensagem enviada.

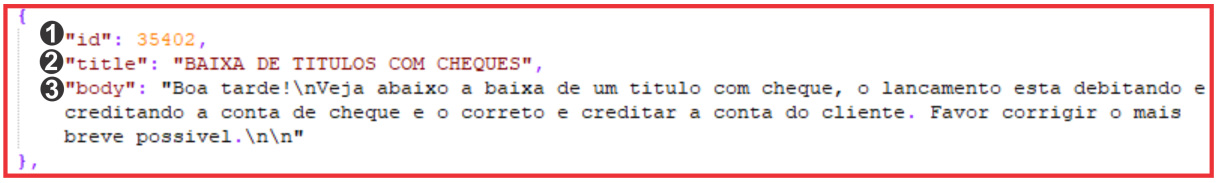

Figura 3.1: Exemplo de documento extraído da base de dados

No total foram extraídos cerca de 30000 chamados. O próximo passo consistiu em realizar o pré-processamento.

Extração de assinaturas dos documentos

Após a extração da base de dados foi iniciada a etapa do pré-processamento nos textos. Primeiramente, observou-se que *tickets* extraídos, por se tratarem da primeira interação e de serem enviados por um cliente da empresa, normalmente continham algum tipo de formalismo como a assinatura do cliente. Como há um grande número de clientes, há também uma grande variedade de assinaturas e isso foi um problema, porque é uma informação que não agrega ao conhecimento a ser obtido, por isso foi necessário a remoção das assinaturas nos *tickets*

O foco nesta etapa foi remover as assinaturas dos textos, cujo padrão é inexistente, sendo encontradas assinaturas diferentes para cada cliente. Para resolver este problema foi criado um *script* em linguagem de programação Python para pré-processar cada chamado buscando padrões de assinatura pré-definidos, com base em diferentes características, que foi executado linha-a-linha, da última para a primeira, onde são buscados padrões que, se encontrados, remove-se a linha atual e as anteriores a ela. Os padrões mostrados na Tabela 3.1 são aqueles comumente encontrados em assinaturas de e-mail.

Padrões de Assinatura
Linha contém endereço de e-mail
Linha contém endereço URL
Linha contém um padrão de nome
Linha contém padrão do tipo "*—*"
O número de linhas em branco é maior que 5
Linha contém saudação do tipo 'Att, Obrigado, Atenciosamente'
Linha contém padrões de CEP ou telefone

Tabela 3.1: Lista de padrões de assinatura

A partir dos padrões de assinatura descritos, o algoritmo foi testado com uma amostra da base de dados original, buscando melhorar sua acurácia. Com isso, é possível listar uma série de padrões que se repetem nos dados, como assinatura de antivírus e textos gerados automaticamente pela sistema.

Entre os padrões encontrados podemos citar também assinaturas automáticas criadas pelo cliente de email utilizado pelo cliente. Para testar a eficácia do algoritmo, foi retirada uma amostra de dez mil chamados e adicionado ao final destes uma *tag* representado a assinatura. O módulo foi executado sobre a amostra e o resultado foi gravado em um novo arquivo com o mesmo formato. Então, um contador foi executado neste novo arquivo e identificou o número

de vezes em que a *tag* ainda aparecia. Em oitenta e oito por cento dos casos as *tags* foram removidas. Ainda há casos em que a assinatura persiste, devendo ser feita uma análise minuciosa dos textos em que a *tag* ainda aparece e procurando novos padrões (como os da Tabela 3.2) para conseguir uma precisão maior na remoção das assinaturas.

Novos Padrões de Assinatura
Primeira linha do arquivo que possui a tag Reunião
Primeira linha do arquivo que possui a tag 'MIME Format'
Primeira linha do arquivo que possui a tag 'Merged Ticket"
Primeira linha do arquivo contento 'This message was created automatically by mail delivery software'
Arquivo contendo '/>' > 3 indicando que é uma página web
Remoção de arquivos com palavras reservadas como 'tecinco.srv.br'
Remoção de linhas que a maioria dos caracteres são caracteres especiais.

Tabela 3.2: Lista de novos padrões de assinatura

Durante a fase de extração das assinaturas foram encontrados padrões nos textos que os tornam inutilizáveis aos próximos passos de pré-processamento. Estes atributos encontrados que invalidam os textos foram listados e, caso encontrados, o texto removido bem como seu id, diminuindo a quantidade de processamento necessário, com o objetivo de não afetar os resultados. A Tabela 3.3 mostra a eficácia da remoção de assinaturas presentes no texto.

Assinaturas removidas	Eficácia
Com os padrões de assinatura	88%
Com os novos padrões de assinatura	98%

Tabela 3.3: Comparação dos resultados da remoção de assinaturas

Realizada a remoção das assinaturas, o tamanho da base de dados diminuiu consideravelmente, tornando o texto mais enxuto. O próximo passo foi tratar o corpo da mensagem removendo palavras vazias e alterando morfologicamente a estrutura do texto. Para realizar essas modificações foram criada algumas estratégias que serão apresentadas na seção a seguir.

3.4 Definição de estratégias de pré-processamento

Como já visto a etapa de pré-processamento é responsável por eliminar ruídos ou conteúdos vazios presentes nos texto. Foi citado também uma variedade de métodos que realizam esta limpeza nos textos, deixando somente informação potencialmente útil para a obtenção de

conhecimento após a aplicação dos algoritmos de mineração de dados. Esses métodos influenciam diretamente sobre a etapa de agrupamento então é preciso avaliar melhores estratégias para realizar a limpeza dos textos.

A definição de estratégias para realizar o processo de pré-processamento teve como objetivo melhorar o resultado na etapa seguinte que foi o agrupamento de textos, uma vez que foi feita a remoção de palavras vazias e alteração da estrutura morfológica do texto, visando melhores resultados, as estratégias foram definidas conforme na Figura 3.2 abaixo.

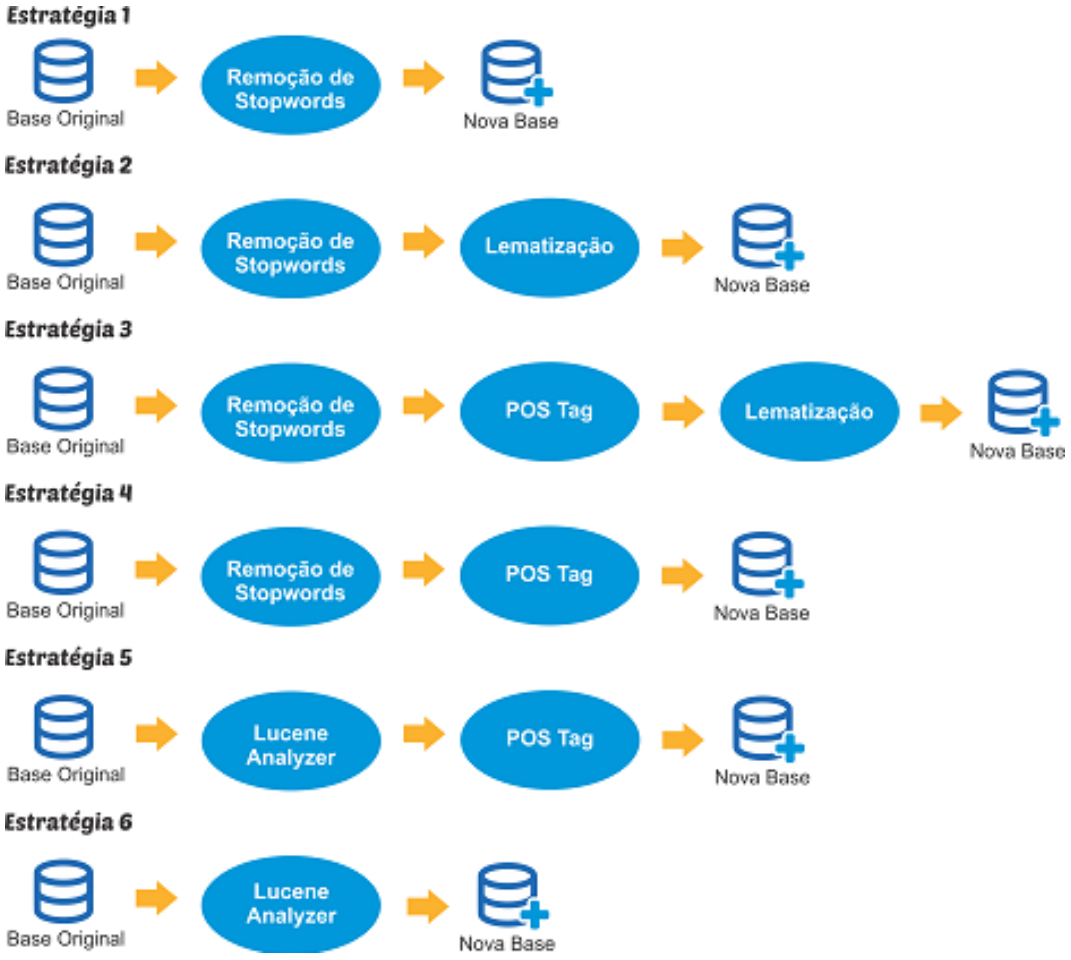

Figura 3.2: Estratégias de pré-processamento

Os resultados das estratégias definidas acima são executadas sobre base de dados original. A base de dados original ou *corpus* original é a coleção de todos os documentos em que as mensagens se encontram. As diferentes estratégias resultam em diferentes saídas ou *corpus* secundários. A ideia de utilizar diferentes estratégias de pré-processamento é justamente pensar em uma melhor acurácia na etapa agrupamento. Estes *corpus* secundários são compostos por

uma variação dos textos dos documentos incorporados a eles, podendo um destes *corpus* gerar melhores resultados que os outros. Assim pode ser definido uma melhor estratégia de pré-processamento para a obtenção de melhores resultados.

3.4.1 Obtenção dos *corpus* secundários

Para a obtenção dos *corpus* secundários foram utilizadas algumas bibliotecas que realizam as tarefas propostas na Seção 2.3.2. Para identificação das partes do discurso (*POS*) foi utilizado a biblioteca *TreeTagger* implementada na linguagem de programação Java. A biblioteca *NLTK*, disponível na linguagem de programação Python, foi utilizada para tokenização, remoção de *stopwords* e conversão de letras maiúsculas em minúsculas. Por fim, a biblioteca *Lucene Analyzer* disponível em Java foi utilizada para a extração de radicais utilizando o método de Porter *(Porter Stem)* para remover inflexões comuns de palavras em português.

3.5 Representação dos textos - *VSM*

O modelo de representação utilizado foi o *VSM* (*vector-space-model*) por ser um dos modelos mais utilizados para representação de documentos em formato textual. Muitas ferramentas usadas em mineração de dados aceitam este formato. Antes do processo de agrupamento é preciso representar os textos em um formato que seja possível calcular a semelhança entre os documentos. Na área de recuperação da informação existem vários modelos para fazer a representação de documentos, sendo que no modelo utilizado, cada documento é visto como um vetor em um espaço multidimensional e cada dimensão é um termo da coleção. Os textos são estruturados em uma *bag of words*, que é uma tabela do tipo documento termo, que relaciona quantas vezes os termos *j* aparecem em *i* documentos. O método utilizado é conhecido como um modelo clássico de representação.

3.5.1 Matriz de semelhanças

Para construção de uma matriz de semelhanças foi utilizado uma medida de similaridade, que calcula o quão iguais dois objetos são. Esta técnica é fundamental para a análise de agrupamentos. A medida de similaridade escolhida é a cosseno, onde o resultado calcula o ângulo cosseno formado entre os vetores de dois documentos. Essa medida não considera a magnitude

dos documentos para computar a proximidade entre eles.

A matriz de semelhanças apresenta os resultados da similaridade de cada *ticket* em uma matriz com dimensão *n*, sendo *n* o número de *tickets* presentes na análise. A Tabela 3.4 mostra a semelhança entre todos os *n* tickets de uma coleção de documentos.

$Ticket/Ticket$	$Ticket_1$	$Ticket_2$	...	$Ticket_n$
$Ticket_1$	1	0.6	...	0.22
$Ticket_2$	-	1	...	0.87
...	...	...	...	...
$Ticket_m$	-	-	...	1

Tabela 3.4: Exmplo de matriz de similaridade

3.6 Agrupamento de dados

Como visto para realizar o agrupamento em bases textuais é necessário convertê-la para um modelo de representação, adaptando os dados para um formato que seja aceito pelos algoritmos de mineração. No entanto, esta conversão pode não ser ainda suficiente para atingir bons resultados na etapa de agrupamento. A avaliação dos resultados gerados pelo agrupamento foram avaliados pelo Coeficiente de silhouette, um critério interno para avaliação de agrupamentos. Em busca de uma acurácia maior na aplicação dos algoritmos de agrupamento, foram adicionadas mais duas técnicas para melhorar a representabilidade dos documentos, a normalização e o *SVD (Single Value Decomposition)*, ambas técnicas descritas a seguir.

3.6.1 Normalização dos dados

Dado o modelo *VSM*, a normalização se refere ao redimensionamento das frequências das palavras de forma que a frequência seja um valor entre 0 e 1 (norma unitária em álgebra linear). Esta abordagem faz parte da etapa de pré-processamento e segundo (BROWNLEE, 2016) é muito útil para matrizes esparsas (grandes quantidades de 0's) que é o caso da tabela *VSM* utilizada. Esta tabela é representada como uma matriz esparsa, nela as linhas são convertidas para uma representação esparsa compactada.

3.6.2 *Single Value Decomposition - SVD*

Para a busca de padrões em textos, estes precisam estar estruturados adequadamente, devido a diversidade e redundância do uso das palavras em coleções textuais. A representação vetorial para a base de dados em estudo possui uma alta dimensionalidade e isto causa impactos negativos, tanto pelo custo computacional exigido quanto para o desempenho da tarefa de agrupamento (LIMA; EUSTÁQUIO; NOGUEIRA, 2017). Visando resolver este problema técnicas de redução de dimensionalidade não supervisionada têm sido empregadas.

A maldição da dimensionalidade é um dos maiores desafios associados a descoberta de conhecimento em textos (ZERVAS; RUGER, 1999). Dada a representação de documentos por vetores de alta dimensionalidade, o agrupamento tende a ficar mais difícil de ser realizado [(LIMA; EUSTÁQUIO; NOGUEIRA, 2017)]. Nestes casos, para quaisquer pares de documentos, as distâncias tentem a ser constantes no espaço vetorial. A Análise Semântica Latente *(Latent Semantic Analisys - LSA)* é uma técnica comumente utilizada para redução de dimensionalidade em textos, visto que consegue identificar bem os conceitos semânticos adjacentes nestes dados (LIMA; EUSTÁQUIO; NOGUEIRA, 2017).

O *LSA* aplica a *SVD* dado como entrada a tabela *VSM* e transforma esta matriz em um espaço "semântico"de baixa dimensionalidade. Particularmente, o *LSA* reduz os efeitos de sinônimos e polissemia (palavras com múltiplos significados), fazendo com que o formato esparso do *VSM* mostre baixa similaridade quando exposto a medidas de distância como a coseno e a euclidiana (MANNING; RAGHAVAN; SCHÜTZE, 2008).

3.6.3 Coeficiente de Silhouette

O coeficiente de Silhouette é um critério interno para avaliação de agrupamentos. Este índice permite avaliar o agrupamento de modo não supervisionado. Um alto resultado do coeficiente de Silhouette está relacionado a um modelo que define bem um grupo (ROUSSEEUW, 1987).

O coeficiente para um elemento é definido na Equação 3.1 com base em duas medidas:

- **a:** a distância média entre um elemento e todos os demais elementos do seu grupo.
- **b:** a distância média deste elemento e todos os elementos do grupo mais próximo.

$$S = \frac{b - a}{max(a, b)} \tag{3.1}$$

Para obter o coeficiente Silhouette para todo o conjunto, faz-se a média de todos os elementos naquele conjunto. O resultado é um valor entre -1 e 1, -1 indicando um agrupamento incorreto e 1 um grupo bem separado, valores próximos de 0 indicam grupos sobrepostos.

3.6.4 Detalhes da implementação

O conteúdo do(s) *corpus* já pré-processados são carregados em um estrutura de dados chamada *data frame* e são extraídas as características desejadas, no caso o texto e o id. O usuário pode informar se deseja uma amostra ou a base de dados inteira. Caso uma amostra seja informada uma amostra aleatória de tamanho N informado pelo usuário é retirada. São encontradas e removidas palavras que se repetem poucas vezes no *corpus*, sendo o número mínimo de repetições informado pelo usuário.

Antes de criar o *VSM* a base de dados passa por um processo de remoção de ruídos, utilizando o algoritmo *HDBSCAN*, uma variação do *DBSCAN*. A opção por esta abordagem se deu uma vez que o agrupamento por densidade é mais sensível a ruídos, podendo detectar e eliminar documentos inconsistentes. São utilizados dois parâmetros no *HDBSCAN* o *min_cluster_size*, que define a quantidade miníma de elementos necessária para formar um grupo, e *min_samples*, que indica o quão conservativo um grupo deve ser. Assim, quanto maior o valor de *min_samples* mais conservativo é o grupo e mais pontos serão declarados como ruídos, sendo os grupos restritos a regiões mais densas. Os valores usados para *min_cluster_size* e *min_samples* são 60 e 1, respectivamente. A escolha destes parâmetros teve o intuito identificar uma quantidade menor ruídos, uma vez que documentos textuais são mais sensíveis a serem classificados como ruídos.

A criação da tabela documento termo ou *VSM* foi criada passando como parâmetro a coluna dos textos do *data frame*, retornando uma matriz X. A frequência desta matriz é normalizada e ocorre a redução de dimensionalidade via *SVD*, tomando-se um número reduzido de componentes.

O agrupamento foi realizado via *K-means* e *MiniBatchKmeans* (*K-Star* não foi usado, pois não apresentou bons resultados nos testes iniciais). Em ambos, o número de grupos k a ser formado varia de 2 a 9, sendo definido como k aquele que apresentar um melhor resultado pela

avaliação do coeficiente de Silhouette. Após a execução do agrupamento os grupos formados são salvos e resgatados posteriormente na implementação do *chatbot*.

3.7 Metodologia do *chatbot*

Chatbot é um agente conversacional que interage com usuários sobre um determinado domínio ou tópico com sentenças de linguagem natural (HUANG; ZHOU; YANG, 2007). O usuário faz uma pergunta ou comentário dentro do seu domínio e o *chatbot* responde. Esta interação é feita por meio da identificação de *Intents* e *Entities*.

Intent é a intenção do usuário ou dos usuários quando interagem com o *chatbot*. Para ilustrar veja a frase abaixo:

"Vai chover hoje?"

Neste caso a *Intent* da frase é saber sobre o clima, ou se vai ou não chover especificamente, então a *Intent* pode ser categorizada como **clima**. A frase acima ainda não é satisfatória para uma boa resposta do *chatbot*. Por exemplo, ele pode querer saber qual a localização do usuário. Nesse contexto aparecem as *Entities* que são valores ou objetos que o *chatbot* irá processar. Para ilustrar, considere a frase:

*"Vai chover hoje em **Cascavel**?"*

Nesta frase o *chatbot* entende que a *Entity* a ser extraída é Cascavel e se trata de uma localização geográfica. Este é um valor variável que identifica a cidade digitada pelo usuário seja ela qual for.

Na hora de construir um *chatbot* é preciso definir *a priori* quais são *Intents* e *Entities* que se deseja extrair dos diálogos. O usuário fornece exemplos de diálogo de como são essas características e o *chatbot* os utiliza para treinar e reconhecer estes padrões usando processamento de linguagem natural. A resposta do chat resulta em uma ação que pode ser uma resposta simples, uma consulta ao banco de dados ou qualquer coisa que possa ser programada.

O *chatbot* foi implementado utilizando um *framework open-source* especifico para criação de *chatbots* o RASA[1], que permite a construção de *chatbots* totalmente funcionais e completos.

[1]Disponível em https://rasa.com/

A escolha por está ferramenta foi devido ao alto nível de personalização que ela proporciona, sendo possível programar as funções que o *chatbot* realiza. As principais ferramentas disponibilizadas pelo RASA, são:

- **Rasa NLU**: Uma solução envolvendo a compreensão de linguagem natural, o qual age sobre a entrada do usuário deduzindo a *Intent* e extraindo as *Entities* disponíveis.
- **Rasa Core**: Uma solução de gerenciamento de diálogos, que constrói um modelo probabilístico que decide que ações tomar baseando-se em diálogos passados.

Para implementação, o RASA necessita de três arquivos que são responsáveis pelo treinamento e modelagem do *bot*, que são:

- **NLU training file**: Contém exemplos de entradas do usuário, juntamente com *Intents* e *Entities* que se encaixam em cada uma destas. Quanto maior for a variedade de exemplos, mais eficaz é o PLN do *bot*.
- **Stories file**: Armazena as histórias com os diálogos que o *bot* utilizará para aprender, para cada história é criado um modelo probabilístico de interações.
- **Domain file**: Aqui são listados todas as *intents*, *entitities* e ações. São adicionados também *templates* de respostas que o *bot* pode tomar.

A Figura 3.3 mostra como os elementos do *chatbot* se relacionam entre si. Os *bots* atuam sobre seu domínio e identificam a intenção do usuário com base na sua entrada. A ação que o *bot* realiza depende exclusivamente da *Intent* que ele identifica e a resposta tem base em um *template* que o usuário define na fase de implementação.

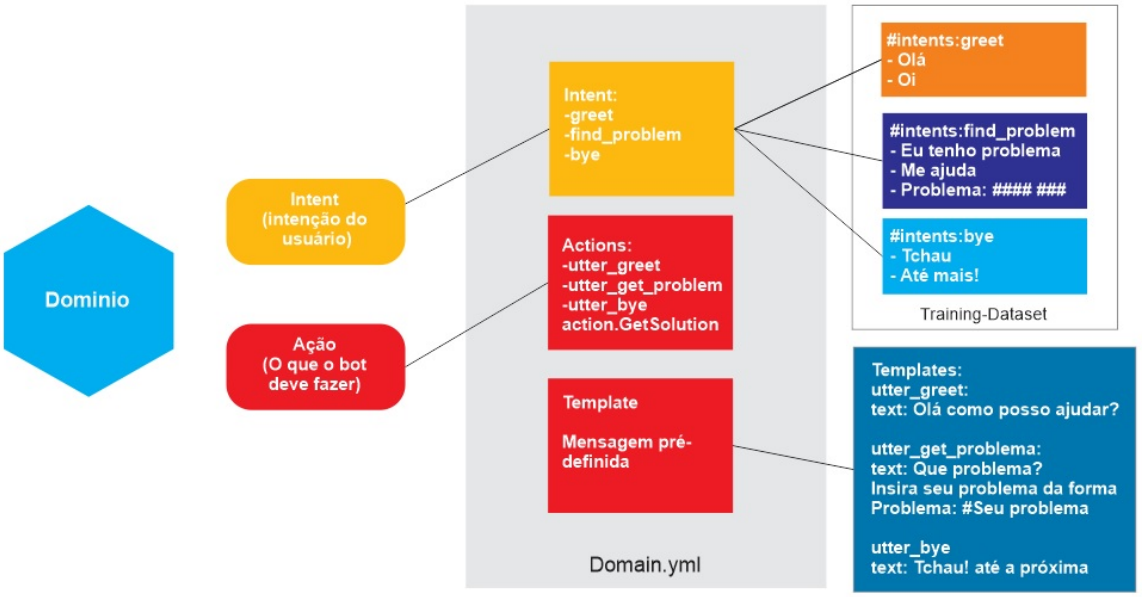

Figura 3.3: Estrutura do *chatbot*

A relação identifica uma *Intent* e realiza uma ação que depende do fluxo de diálogo que foi estabelecido na implementação. Este fluxo é treinado continuamente e o *bot* aprende com interações passadas, para identificar novos fluxos de diálogo.

A próxima Seção apresenta o protótipo do *chatbot*, bem como são mostrados e discutidos os resultados obtidos no agrupamento.

Capítulo 4

Resultados e discussão

Os resultados gerados incluem os novos *corpus* criados a partir das estratégias de pré-processamento. Estes corpus são usados na etapa de agrupamento por meio de diferentes abordagens gerando múltiplos resultados que, comparados pelo coeficiente de Silhouette, mostram qual o melhor *corpus* e abordagem utilizar para o agrupamento e no protótipo do *chatbot*. O *chatbot* vai receber deste corpus os grupos gerados e quando interagir com usuário e identificar o problema, ele vai ser comparado com estes grupos, atribuindo-o aquele mais próximo. O *chatbot* vai retornar como resultado uma solução baseada nas palavras mais frequentes daquele grupo, deixando como opção (caso necessário), retornar alguns documentos pertencentes ao grupo no qual o problema foi atribuído.

Foram gerados seis *corpus* secundários a partir dos *corpus* original através das estratégias de pré-processamento definidas na Seção 3.4. Estes *corpus* gerados são mais objetivos e pertinentes para a etapa de agrupamento, pois foram removidos palavras sem significado, assinaturas e documentos sem relevância para o processo de extração de conhecimento.

4.1 Resultados do agrupamento

Foram aplicados os algoritmos de agrupamento sobre os *corpus* para definir qual a melhor estratégia de pré-processamento, qual a melhor abordagem para o agrupamento e também o algoritmo de agrupamento mais eficaz. O algoritmo de agrupamento *K-star* foi aplicado aos diferentes *corpus*. No entanto, a quantidade de grupos gerados foi excessivamente alta, resultando em muitos grupos compactos sendo, portanto, de difícil interpretação.

Como o desempenho do *K-star* não foi satisfatório relativo à quantidade de grupos forma-

dos, se tornou necessário avaliar outros algoritmos e diferentes abordagens para formação dos grupos. Os resultados destacados nesta seção foram obtidos utilizando os algoritmos *K-means* e *MiniBatchKmeans*.

A amostra utilizada foi de dez mil documentos, sendo limitado este número devido a capacidades de processamento disponível. A amostra foi extraída aleatoriamente dos *corpus* secundários. Os testes avaliaram um valor de k entre 2 e 9, buscandoo *k* que apresenta o melhor resultado. Foram feitas 10 execuções de cada abordagem para todos os *corpus*, o resultado é a média dessas execuções.

4.1.1 Abordagens de agrupamento

Visando a eficácia da metodologia aplicada, foram adotadas algumas abordagens para realizar o agrupamento. Essas abordagens, listadas na Tabela 4.1.1 inferem na variabilidade de técnicas empregadas, sendo que essas técnicas dizem respeito a utilização da frequência relativa ou frequência inversa, eliminação ou não de ruídos, número de componentes do *SVD* (3 ou 5) e utilização de unigramas ou bigramas.

Abordagem	Freq. Inversa	Outliers	componentes SVD	*Ngram*
1	Não	Sim	5	Unigrama
2	Não	Não	5	Bigrama
3	Sim	Não	5	Unigrama
4	Não	Não	3	Unigrama
5	Não	Sim	3	Unigrama

Tabela 4.1: Abordagens utilizadas no agrupamento

Na Tabela 4.2 são mostrados os resultados do algoritmo *K-means* para todas as abordagens. Comparando-se as abordagens, verifica-se vantagem da abordagem 4 em relação as abordagens 2 e 3 para a maioria dos *corpus*, a abordagem 5 foi melhor para quase todos os *corpus*. Assim, a utilização de 3 componentes no *SVD* melhorou a qualidade do agrupamento em relação a utilização de 5 componentes. A utilização da remoção de ruídos não melhorou significamente a qualidade do agrupamento piorando na maioria dos cados.

Dentre os melhores resultados para cada abordagem o *corpus* quatro foi o que apresentou o melhor resultado para todas as abordagens.

Corpus	Abordagem 1	Abordagem 2	Abordagem 3	Abordagem 4	Abordagem 5
Corpus 1	0.437	0.433	*0.516*	0.508	0.512
Corpus 2	0.468	0.449	0.418	0.502	*0.507*
Corpus 3	0.479	0.467	0.425	*0.573*	0.573
Corpus 4	**0.557**	**0.577**	**0.517**	**0.661**	***0.663***
Corpus 5	0.507	0.518	0.459	0.598	*0.602*
Corpus 6	0.329	0.330	0.383	0.422	*0.424*

Tabela 4.2: Resultado médio de dez execuções obtidos pelo *K-means*

A Tabela 4.3 refere-se aos resultados do algoritmo *MiniBatchKmeans*. As abordagens cinco e quatro apresentaram os melhores resultados. O corpus quatro apresentou o melhor resultado.

Corpus	Abordagem 1	Abordagem 2	Abordagem 3	Abordagem 4	Abordagem 5
Corpus 1	0.440	0.425	0.485	*0.514*	0.503
Corpus 2	0.476	0.425	0.412	0.495	*0.497*
Corpus 3	0.480	0.461	0.435	*0.578*	0.571
Corpus 4	**0.548**	**0.584**	**0.534**	**0.663**	***0.666***
Corpus 5	0.510	0.515	0.450	0.605	*0.607*
Corpus 6	0.323	0.327	0.426	*0.426*	0.424

Tabela 4.3: Resultado médio de dez execuções obtidos pelo *MiniBatchKmeans*

O algoritmo que apresentou o melhor resultados para a maioria das abordagens foi o *MiniBatchKmeans*. Apesar da avaliação pelo Coeficiente de Silhouette não ser considerada a ideal, podendo este fato atribuído a diversos fatores, como por exemplo, as estratégias de pré-processamento utilizadas. O *MiniBatchKmeans* foi utilizado para gerar os grupos que apresenta as palavras frequentes na Tabela 4.4.

Grupos	Palavras frequentes
1	sistema, erro, nota, verificar, esta, nf, notas, retorno, mensagem, emitir, fazer, imagem, cliente, conseguindo, problema, dando, entrada, aparece, servico, tela, valor, tcar, consigo, saber, ajuda, porem, arquivo, aguardo, relatorio
2	erro, verificar, nf, dando, esta, imagem, retorno, mensagem, fazer, arquivo, aparece, sistema, notas, emitir, tela, aguardo, cliente, problema, aparecendo, apresentando, servico, tcar, deu, porem, gentileza, hora, relatorio, os, ocorrendo, ordem
3	erro, verificar, dar, estar, nf, fazer, ser, aparecer, imagem, emitir, mensagem, retorno, poder, ocorrer, tentar, apresentar, cliente, gerar, tela, arquivo, problema, servico, aguardar, conseguir, algum, relatorio, entrada, ficar, cancelar, valor
4	nota, emitir, notas, esta, retorno, verificar, entrada, fazer, cliente, ajuda, servico, conseguindo, cancelar, valor, nf, problema, sistema, consigo, dar, emissao, auxilio, aparece, pra, porem, mensagem, imagem, pecas, mesma, devolucao, itens
5	nota, erro, verificar, emitir, esta, nf, retorno, notas, dando, imagem, fazer, entrada, cliente, servico, ajuda, mensagem, aparece, problema, conseguindo, cancelar, valor, consigo, aguardo, porem, tela, emissao, dar, arquivo, auxilio

Tabela 4.4: Palavras mais frequentes para cada grupo gerado

A Tabela 4.4 mostra para cada grupo as trinta palavras mais frequentes. Essas palavras permitem avaliar o tema que é tratado em cada grupo gerado. Esses temas podem ter ou não relação entre si, além de poder utilizar estas palavras para incrementar a lista de *stopwords* identificando novas palavras sem significância para o agrupamento. Também pode-se utilizar como sinônimos algumas palavras com o mesmo significado que aparecem nos grupos formados como nf e nota, que se remetem a um mesmo significado nota fiscal.

4.2 Sistema proposto

O *corpus* que apresentou a melhor acurácia nos testes foi o *corpus* 4. Os grupos formados por ele foram utilizados para o modelo de recuperação implementado no *chatbot*.

Os grupos são formados por coleções de documentos e, dada a estrutura do algoritmo, os documentos próximos do centróide são mais representativos e isso fornece uma importância maior a eles. Considerando também as frequências das palavras do centróide é possível dizer quais são as mais frequentes dentro de um grupo, atribuindo de certa forma um peso maior a elas. Compreendendo esta estrutura foi possível propor um modelo para o *chatbot*.

O *chatbot* implementado neste trabalho classifica novos documentos a um grupo já definido. O novo documento que se deseja categorizar é informado pelo usuário na forma de diálogo através do *chatbot*, o qual retorna o grupo em que a distância para seu centroide é menor. O usuário tem como resposta as palavras mais frequentes daquele grupo, tendo uma noção do assunto que foi abordado nos documentos daquele grupo.

4.2.1 Treinamento do *chatbot*

O treinamento do *chatbot* é feito de duas maneiras: pré-definido por meio de exemplos fornecidos na implementação, ou continuamente conforme os diálogos com o usuário. Da segunda forma o chatbot armazena os diálogos em um arquivo chamado *stories* e o utiliza para criar novos fluxos de dialogo.

A Figura 4.1 demonstra a forma como que o *chatbot* aprende para uma entrada do usuário. O fluxo retrata uma interação entre usuário (de vermelho) e *bot* (de azul), a seguir é posto em detalhes o resultado desta interação.

1. O *bot* espera uma ação do usuário, ou um dialogo inicial.

2. O *bot* reconhece a *Intent* como *greet* de comprimento, o percentual de certeza é de setenta e sete por cento.

3. Acontece uma resposta do *bot*, disparada pela *Intent* encotrada, e definida no template. O *bot* espera uma nova entrada do usuário.

4. A *Intent* identificada para este novo problema é do tipo *find_problem* com sessenta e cinco por cento de certeza, significando que existe um problema em questão.

5. O *bot* especifica o modo de como espera receber o problema e espera uma nova entrada.

6. O *bot* identifica novamente a *Intent find_problem* com oitenta e cinco por cento de certeza, e identifica uma *Entity problem*, encontrando assim o problema especificamente na entrada do usuário como sendo [erro, comprovante, nota].

7. A identificação da *Entity* dispara a ação *get_solution*. A *entity* é passada como parâmetro, e são exibidos as palavras mais frequentes do grupo em que é classificada a entrada.

8. Por fim o usuário se despede, e o *bot* confirma a ação com setenta e cinco por cento de acurácia como ilustra na Figura 4.1.

Figura 4.1: Treinamento do chatbot

4.2.2 Implantação do *chatbot*

O ambiente escolhido para integrar o *chatbot* foi o Slack, que é um ambiente muito utilizado em empresas da área de tecnologia, tendo como objetivo melhorar a comunicação interna e integrar o time. Essas características justificam a implantação do *bot* nesta plataforma, uma vez que várias pessoas podem interagir com ele simultaneamente.

A comunicação do *bot* com o servidor é feita via *webhook*, fazendo com que a troca de informações ocorra em tempo real entre *bot* e usuário. Na Figura 4.2 é possível visualizar a aplicação.

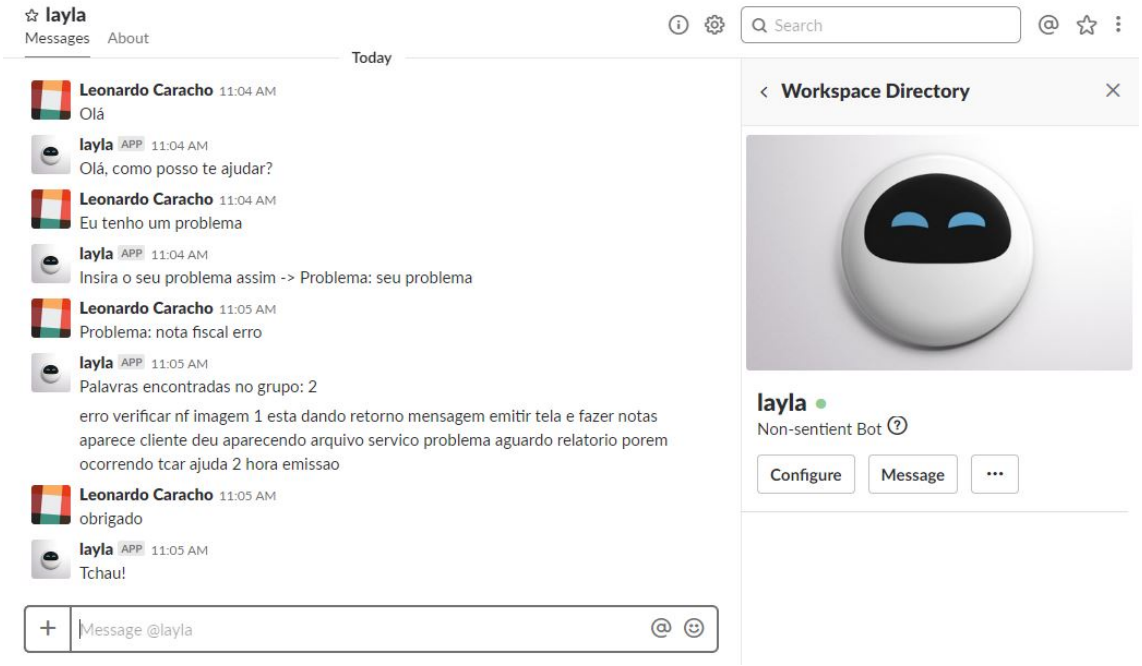

Figura 4.2: *Chatbot* em funcionamento na plataforma Slack

A implementação do *bot* no Slack, apresenta um visual intuitivo e amigável, isto e o modo de como é feita integração são algumas das vantagens da plataforma utilizada. Cabe ainda dizer que o RASA permite que o *bot* seja integrado a qualquer outra plataforma. Neste capítulo foi apresentado o protótipo do *bot* e das interações que ele proporciona, no próximo é discutido algumas formadas de futuramente desenvolver novas funcionalidades, e melhorar as já implementadas.

Capítulo 5

Considerações finais

Este capítulo tem como objetivo apresentar uma visão geral acerca do trabalho desenvolvido, discutindo alguns pontos importantes observados nos experimentos realizados. Também são apresentadas propostas de trabalhos futuros e melhorias no sistema proposto.

5.1 Principais considerações

Na etapa de revisão bibliográfica podê se concluir que o *MiniBatchKmeans* é o algoritmo mais utilizado para o agrupamento de textos e neste trabalho o mesmo também apresentou bons resultados dado o critério de avaliação utilizado. A aplicação do *chatbot* para a recuperação da informação é uma proposta original dada as necessidades levantadas na empresa. Este trabalho, portanto, pode ser utilizado como uma referência para novos métodos de recuperação da informação.

Uma das tarefas mais custosas foi o pré-processamento. A grande quantidade de dados da base original e o formato não estruturado dos dados dificultou a limpeza dos textos, uma vez que os mesmos não possuem um padrão e podem conter características que os invalidam. Identificar essas características e removê-las também poderiam eliminar conteúdo útil dos textos. Por esta razão foram utilizadas diferentes estratégias para o pré-processamento dos textos.

A estratégia que resultou nos melhores resultados foi a que utiliza a remoção de *stopwords* e remoção de advérbios. O idioma dos textos também tem influência direta no pré-processamento, uma vez que os algoritmos em grande maioria estão otimizados para o inglês, o que não é o caso deste trabalho.

Na etapa de agrupamento o uso do *DBSCAN* não se mostrou bom para identificar e remover

ruídos nos textos. Os grupos gerados contém informações úteis como, por exemplo, a possibilidade de identificar as palavras mais frequentes de um grupo específico ou os documentos mais representativos daquele grupo.

O *chatbot* utiliza os grupos gerados para fazer um modelo de recuperação da informação, permitindo o usuário resgatar, para uma nova entrada, as palavras mais frequentes do grupo a que ela pertente ou os documentos similares. Este modelo foi criado seguindo fluxo de diálogo, onde o *bot* interpreta a conversa através de um modelo probabilístico criado através de algorítimos de processamento de linguagem natural do próprio *framework* utilizado para criação do *chatbot*. O modelo criado pode ser utilizado para ser aprimorado criando novas linhas de diálogo ou melhorando a precisão para extração de *Entities* e *Intents*.

5.2 Trabalhos futuros

Em relação a qualidade do agrupamento, sugere-se avaliar possíveis causas e necessidades para melhorar a eficácia. Dentre elas: geração de novos *corpus* utilizando outras estratégias; utilização de diferentes abordagens, incluindo etapas adicionais no pré-processamento; avaliação de outros algoritmos de agrupamento; dentre outros.

O *chatbot* desenvolvido neste trabalho se apresenta como um protótipo inicial, podendo ser aprimorado, adicionando-se novos *features* ou técnicas de mineração de dados como regras de associação para descobrir problemas que ocorrem em comum dentro do conjunto de dados. Melhorias ou novas funcionalidades que podem ser adicionadas são:

- Criar novos ramos de diálogo, ampliando o domínio do *chatbot*.
- Permitir ao usuário realizar outros tipos de consultas sobre os dados.
- Inserir regras de associação para determinar problemas que ocorrem em comum.
- Implantar e integrar o *chatbot* com o serviço de suporte da empresa.
- Validar o *chatbot* em conjunto com o serviço de suporte.

Referências Bibliográficas

ARANHA, C.; PASSOS, E. A tecnologia de mineração de textos. *Revista Eletrônica de Sistemas de Informação*, v. 5, n. 2, 2006.

BROWNLEE, J. Machine learning mastery with python. *Machine Learning Mastery Pty Ltd*, p. 100–120, 2016.

CORRÊA, A. C. G. et al. Recuperação de documentos baseados em informação semântica no ambiente ammo. Universidade Federal de São Carlos, 2003.

CUTTING, D. R. et al. Scatter/gather: Browsing a cluster-based large document approach collections to scatter/gather. In: *Sigir*. Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Copenhagen, Denmark, June 21–24).: ACM, 1992. v. 92, p. 318–329.

EBECKEN, N. F. et al. Mineração de textos. *Sistemas inteligentes: fundamentos e aplicações. São Carlos: Manole*, p. 337–370, 2003.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. Portland, Oregon: AAAI Press, 1996. v. 96, n. 34, p. 226–231.

EVERITT, B. S.; LANDAU, S.; LEESE, M. Cluster analysis–arnold. *A member of the Hodder Headline Group, London*, p. 429–438, 2001.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI magazine*, v. 17, n. 3, p. 37, 1996.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, ACM, v. 39, n. 11, p. 27–34, 1996.

FELDMAN, R.; SANGER, J. et al. *The text mining handbook: advanced approaches in analyzing unstructured data*. New York: Cambridge university press, 2007.

GOEBEL, M.; GRUENWALD, L. A survey of data mining and knowledge discovery software tools. *ACM SIGKDD explorations newsletter*, ACM, v. 1, n. 1, p. 20–33, 1999.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. Cluster validity methods: part i. *ACM Sigmod Record*, ACM, v. 31, n. 2, p. 40–45, 2002.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. Clustering validity checking methods: part ii. *ACM Sigmod Record*, ACM, v. 31, n. 3, p. 19–27, 2002.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. New York: Elsevier, 2011.

HUANG, J.; ZHOU, M.; YANG, D. Extracting chatbot knowledge from online discussion forums. In Proc. of IJCAI, v. 7, p. 423–428, 2007.

LIMA, B.; EUSTÁQUIO, F.; NOGUEIRA, T. Influência de técnicas não-supervisionadas de redução de dimensionalidade para organização flexível de documentos (the unsupervised dimensionality reduction weight on flexible document organization)[in portuguese]. In: *Proceedings of the 11th Brazilian Symposium in Information and Human Language Technology*. [S.l.: s.n.], 2017. p. 112–121.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. Matrix decompositions and latent semantic indexing. *Introduction to Information Retrieval*, Cambridge University Press Cambridge, UK, p. 403–417, 2008.

MORAIS, E. A. M.; AMBRÓSIO, A. P. L. Mineração de textos. *Relatório Técnico–Instituto de Informática (UFG)*, 2007.

PIATETSKY-SHAPIRO, G.; FAYYAD, U.; SMITH, P. From data mining to knowledge discovery: An overview. *Advances in knowledge discovery and data mining*, v. 1, p. 35, 1996.

PINTO, D. et al. Buap: Performance of k-star at the inex'09 clustering task. In: SPRINGER. *International Workshop of the Initiative for the Evaluation of XML Retrieval*. Woodlands of Marburg, Ipswich, Queensland, Australia., 2009. p. 434–440.

REZENDE, S. O.; MARCACINI, R. M.; MOURA, M. F. O uso da mineração de textos para extração e organização não supervisionada de conhecimento. *Revista de Sistemas de Informação da FSMA*, n, v. 7, p. 7–21, 2011.

ROBERTSON, S. E.; WALKER, S. On relevance weights with little relevance information. In: ACM. *ACM SIGIR Forum*. New York, 1997. v. 31, n. SI, p. 16–24.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, Elsevier, v. 24, n. 5, p. 513–523, 1988.

SANDER, J. et al. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, Springer, v. 2, n. 2, p. 169–194, 1998.

SHIN, K.; HAN, S. Fast clustering algorithm for information organization. In: SPRINGER. *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, Heidelberg, 2003. p. 619–622.

STEINBACH, M. et al. A comparison of document clustering techniques. In: BOSTON. *KDD workshop on text mining*. Boston, 2000. v. 400, n. 1, p. 525–526.

TAN, A.-H. et al. Text mining: The state of the art and the challenges. In: SN. *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*. Beijing, 1999. v. 8, p. 65–70.

WIVES, L. K. Tecnologias de descoberta de conhecimentos em textos aplicadas à inteligência competitiva. 2002. *Cited on*, p. 86, 2002.

WIVES, L. K. *Utilizando Conceitos como descritores de Textos para o processo de identificação de conglomerados (clustering) de documentos. Porto Alegre: 2004. 136 f*. Tese (Doutorado) — Tese (Doutorado em Ciência da Computação)-Universidade Federal do Rio Grande do Sul, 2004.

YE, Z. et al. Part-of-speech tagging based on dictionary and statistical machine learning. In: IEEE. *2016 35th Chinese Control Conference (CCC)*. China, 2016. p. 6993–6998.

ZERVAS, G.; RUGER, S. M. The curse of dimensionality and document clustering. IET, 1999.