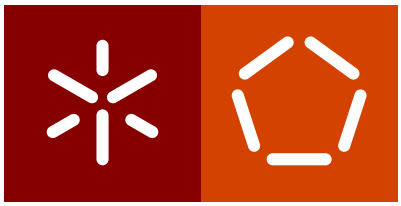
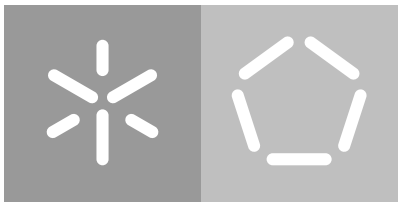


Universidade do Minho
Escola de Engenharia
Departamento de Sistemas de Informação

Luís Ricardo Tibo Machado dos Santos

Text Mining aplicado ao serviço ao cliente em telecomunicações

Abril 2020



Universidade do Minho
Escola de Engenharia
Departamento de Sistemas de Informação

Luís Ricardo Tibo Machado dos Santos

Text Mining aplicado ao serviço ao cliente em telecomunicações

Dissertação de Mestrado
Mestrado em Gestão de Sistemas de Informação

Dissertação supervisionada por
Paulo Alexandre Ribeiro Cortez
Carlos Miguel Silva Couto Pereira

Abril 2020

ACRÓNIMOS

CCW	Class Confidence Weights
IoT	Internet of Things
IR	Information Retrieval
KEA	Keyphrase Extraction Algorithm
kNN	k-Nearest Neighbour
ML	Machine Learning
NER	Named Entity Recognition
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
PMML	Predictive Model Markup Language
SEMMA	Sample-Explore-Modify-Model-Assess
SVM	Support Vector Machines
VSM	Vector Space Model
NPS	Network Promoter Score

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUÇÃO

1.1 ENQUADRAMENTO E MOTIVAÇÃO

O presente trabalho e dissertação foram desenvolvidos no âmbito de um estágio curricular com a empresa NOS. A parceria entre a Universidade do Minho permitiu a combinação do conhecimento dos tutores universitários com o conhecimento e as tecnologias desenvolvidas e factor de inovação da NOS, para conduzir este trabalho para o caminho certo.

O conceito de informação deriva do latim e significa um processo de comunicação ou algo relacionado com comunicação [Zha88], mas na realidade existem muitas e variadas definições de informação, cada uma mais complexa que outra. Podemos também dizer que Informação é um processo que visa o conhecimento, ou, mais simplesmente, "Informação é tudo o que reduz a incerteza... Um instrumento de compreensão do mundo e da ação sobre ele" [Zor95]. Mais de 2.5 triliões de *Bytes* são gerados diariamente, dados esses que são gerados automaticamente e continuamente, e segundo a tendência atual é de continuar a aumentar juntamente com o crescimento da IoT (*Internet of Things*) [Mar18], este crescimento pode ser considerado quase exponencial nos últimos anos. A informação é cada vez mais um fator decisivo nas organizações por ser um recurso indispensável tanto no contexto interno como no contexto interno das mesmas como no relacionamento com o exterior. Torna-se importante para qualquer organização conseguir analisar os dados de uma forma inteligente e acima de tudo funcional, com o objetivo de representar os dados de formas que ajudem à sua posterior análise, revelar várias características importantes para a tomada de decisão e permitir identificar fontes de padrões e variações nos dados de uma maneira em que seja possível conseguir explicar a origem dessas mesmas variações. Muita desta informação vem através da forma de dados não estruturados (cerca de 85%), a maior parte em forma de texto, e de forma a ser possível analisar e extrair elementos essenciais desses dados não estruturados são aplicadas técnicas de *Text Mining*. Este projeto de dissertação envolve área de *Text Mining* aplicada ao serviço de atendimento ao cliente em telecomunicações. Pretende-se explorar as várias técnicas de *Text Mining* e de processamento de linguagem natural. Para isso, será utilizada a metodologia CRISP-DM. Os dados serão fornecidos

pela empresa em questão, tendo como caraterísticas terem um grande volume e serem maioritariamente dados não estruturados.

1.2 OBJETIVOS E RESULTADOS ESPERADOS

A definição clara de objetivos e resultados esperados é fulcral para o desenvolvimento e sucesso de um projeto. Nesse sentido definimos o principal objectivo deste trabalho consiste em extrair informação relevante de dados não-estruturados fornecidos pela NOS. Técnicas de processamento de *Text Mining* e processamento linguagem natural serão usadas para extracção de entidades, dos principais tópicos e análise de sentimento. É ainda esperado um desenvolvimento de um sistema de processamento de linguagem natural que permita a extracção e análise da informação relevante de dados não-estruturados. É também necessário definir as ferramentas tecnológicas que irão ser utilizadas durante a dissertação, bem como o planeamento das tarefas a realizar.

1.3 CONTRIBUIÇÕES

Após a finalização deste projeto de dissertação espera-se um bom levantamento estudo do estado da arte que permita perceber que conceitos relativos a este projeto que foram utilizados. Para além deste documento vai ser desenvolvido um sistema de processamento de linguagem natural que permita uma análise e compreensão de dados não-estruturados, uma extração dos dados através de métodos como *stemming*, *lemmatization*, NER entre outros. O sistema permitirá ainda utilizar modelos capazes de analisar dados maiores e mais complexos, e entregar resultados mais rápidos e precisos com o mínimo de erro. Alguns dos modelos a serem utilizados serão *Decision Trees*, *SVM*, *Naive Bayes*, kNN, entre outros. Com estes modelos espera-se tirar conclusões relevantes e precisas, avaliando a precisão e desempenho através de métricas pré-definidas, relativamente aos principais tópicos e efetuar uma análise de sentimento.

1.4 ESTRUTURA DO DOCUMENTO

O presente documento é constituído por quatro capítulos distintos. Em primeiro lugar, o Capítulo 1 (Introdução) que é composto pelo Enquadramento e Motivação, Objetivos, Contribuições e a Estrutura do documento desta dissertação. Em seguida, no Capítulo 2(*State of the Art*) já são introduzidos conceitos relevantes para a dissertação. Foi efetuado o levantamento de vários conceitos relativos às áreas de *Text Mining*, Pré-Processamento que terá dois subcapítulos acerca de *Data Cleansing* e *Tokenization* e referir o seu conceito e os seus diferentes âmbitos. Este segundo capítulo sendo o mais extenso é ainda composto por

mais conceitos como *Keyword Extration*, *Text Summarization*, NER, *Topic Modeling*, Análise de sentimentos que terá como subcapítulos *Supervised Learning*, Classificação, Modelos ML e ainda Métricas de Avaliação. Por fim, na última secção deste capítulo serão identificadas as tecnologias necessárias para o desenvolvimento deste projeto de dissertação bem como o seu propósito. Já no Capítulo 3, será abordada a Metodologia a utilizar ao longo do trabalho e as suas componentes como *Design Science Research*, CRISP-DM e será ainda elaborada uma descrição do planeamento das tarefas do projeto e a estimação de tempos de execução. Por fim, no quarto e último capítulo será apresentada a conclusão deste projeto e será ainda descrito o *Future Work*.

2

ESTADO DA ARTE

2.1 *text mining*

Uma vez que a comunicação escrita é algo de natureza humana, existem certas nuances que apesar de serem bastante triviais para uma pessoa, um algoritmo computacional apresentará algumas dificuldades em interpretá-las corretamente [GL09]. Mas, por outro lado os computadores têm outras capacidades que para um humano seria impossível. Seria impossível analisar grandes volumes de textos sem ajuda de um computador. E o *Text Mining* surge um bocado nessa necessidade, na necessidade de extrair padrões e conhecimento de estruturas de dados não estruturadas ou semiestruturadas, tais como documentos de texto. *Data Mining* (DM) visa extrair conhecimento útil (por exemplo, padrões ou tendências) de dados não estruturados [WFH]. *Text Mining* é um tipo particular de DM que permite extrair padrões e conhecimento, mas aplicada ao caso específico de ser em documentos de texto. *Text-mining* refere-se ao processo de retirar informação com grande qualidade sobre texto e o principal objetivo é essencialmente para tornar texto numa análise de dados via *Natural Language Processing*(NLP). É importante ainda ver todas as aplicações de *Text Mining*, mesmo em outras áreas como *Information Extraction*, *Information Retrieval* e *Machine Learning*. Text Mining não é apenas uma ferramenta que permite pesquisar, também consegue determinar certos atributos de certas palavras, por exemplo saber que a palavra "cão" é um animal, e que a palavra "sentar" é uma ação. Permite-nos ainda, entre muitas outras coisas conseguir saber quais são os cães que se sentam, entre milhões de artigos. E por isso, o conjunto de técnicas a utilizar nem sempre é a mesma, com o desenvolvimento *web* e surgiram também casos específicos para a aplicação de *Text Mining*. Em 2013 He, Zha, & Li realçam que, com a crescente presença das empresas nas plataformas de *social media* existe cada vez mais a necessidade de realizar uma recolha de dados nas suas redes sociais com o intuito de descobrir padrões presentes no texto. [OCA17] propuseram uma metodologia capaz de prever vários indicadores financeiros relacionados com o mercado da bolsa. A metodologia utiliza a polaridade dos sentimentos de *tweets*. Os mesmo autores, num outro estudo, utilizam dados extraídos da plataforma *StockTwits* para prever as variáveis dos mercados financeiros [OCA13] através de modelos de regressão.

2.2 PRE-PROCESSAMENTO

O pré-processamento de um texto prepara-o para processamento posterior. Os passos típicos incluem a *tokenization*, a remoção de *stop-words* (por exemplo, o, é, a) e a *lemmatization* , ou seja, a redução de diferentes formas morfológicas de uma palavra para uma forma de raiz (por exemplo, tanto o cortado como o cortador são lemmatizados para cortar).

2.2.1 *Data Cleaning*

Diz-se que *Data scientists* dispendem 80% do seu tempo a realizar limpeza e manipulação de dados e apenas 20% a analisar estes dados. *Data Cleansing* é um processo cada vez mais relevante nos dias de hoje devido à quantidade de informação que é guardada diariamente e porque é necessário analisar essa informação de modo a obter resultados verdadeiros e que permitem retirar conclusões. *Data Cleansing*, também chamado de *Data Cleaning* vem de uma certa forma responder a esta necessidade. *Data Cleansing* é o processo de preparação de dados para análise dos mesmos. Este processo atravessa duas fases, a deteção e correção ou transformação dos dados. Para esta análise é necessário detetar previamente que tipo de erros e inconsistências existem e proceder à sua remoção ou modificação, como por exemplo de dados que estejam incorretos ou incompletos, dados irrelevantes, duplicados ou formatados incorretamente, tudo com fim de melhorar a qualidade dos dados. Estes dados geralmente não são uteis quando se trata de analisar dados, podendo até dificultar o processo de análise ou fornecer resultados imprecisos. Quando múltiplas fontes de dados necessitam de ser integradas, este processo torna-se ainda mais relevante uma vez que um dos objetivos é criar conjuntos de dados que seja uniforme de modo a que as ferramentas de análise de dados possam facilmente encontrar os dados para consulta e proceder a uma fácil análise. Mas como os dados provêm de várias fontes, muitas vezes, contém dados redundantes em diferentes representações. Algumas técnicas que podem ser usadas para resolver alguns dos problemas que podem surgir são:

1. Remoção de espaços;
2. Tratar de células em branco;
3. Conversão de números guardados como texto e texto como número;
4. Remoção de duplicados;
5. Texto para minúsculas ou maiúsculas (*Proper Case*);
6. Remover formatações;

2.2.2 *Tokenization*

Palavras, e *tokens* em geral, são os principais blocos de construção em quase todas as teorias linguísticas [Gaz85][Hud84] e sistemas de processamento linguístico [All95][Gro86]. O processo de mapeamento de frases de *strings* de caracteres para *strings* de palavras, é o passo inicial no NLP [Sch92]. *Tokenization* é o nome que se dá ao ato de quebrar uma sequência de *strings* de palavaras em pedaços mais pequenos como palavras, *keywords*, *strings* ou outros elementos chamados *tokens*. Um token pode ser uma palavra individual, um número ou até uma frase inteira. No processo de *tokenization*, alguns caracteres como sinais de pontuação são geralmente descartados, e os *tokens* tornam-se futuramente no input para outro processo, como por exemplo *Text Mining*. Quando falamos de tokenization, em NLP, o primeiro passo é identificar os *tokens*, ou aquelas unidades básicas que não precisam ser decompostas num processamento subsequente, antes de qualquer tipo de processamento. O processo de *tokenization* faz esta tarefa através da localização de *word boundaries*.*Word boundaries* são o ponto final de uma palavra e o ínicio da palavra seguinte, este tipo de *tokens* é muito útil para encontrar vários padrões e é ainda considerado um "passo base" para *stemming* e *lemmatization*. Por norma os *tokens* são separados por espaços em branco, pontos de pontuação ou quebras de linha. Existe ainda caso que o espaço em branco ou pontos de pontuação possam ou não ser incluídos como separadores, tudo depende da necessidade. Os *tokens* em si também podem ser separadores. Na maioria das linguagens de programação, os identificadores, que são um tipo de *token*, podem ser colocados juntamente com operadores aritméticos sem espaços brancos. Embora a primeira vista possa parecer que tudo seria apenas um *token* a gramática da linguagem realmente considera o operador como um separador, deste modo mesmo que vários *tokens* estejam agrupados podem ser separados através do operador matemático. Como todas as processos baseados em NLP os diferentes idiomas apresentam diversas barreiras. E *tokenization* apresenta o mesmo tipo de problemas em vários idiomas incluindo o japonês, coreano, alemão, inglês, português entre outros. Estes problemas existem em vários meios, como a fala contínua e *cursive handwriting*, e em numerosas aplicações, como tradução, reconhecimento, indexação e revisão. O uso de abreviatura é outro problema pois pode levar o *tokenizer* a detetar um limite onde não há nenhum. Na maioria das linguagens que utilizam o alfabeto latino e linguagens de programação é bastante máis fácil, mas existem sempre contradições: palavras hifenizados, *emoticons*, etc. Um exemplo clássico é "*New York-based*", onde um *tokenizer* mais ingénuo pode quebrar a *string* no espaço mesmo que seja melhor efetuar a quebra no hífen.

2.2.3 *Word Embeddings*

A tarefa de representar palavras e documentos é uma grande parte das tarefas de NLP, e a modelagem estatística da linguagem (*Statistical Language modeling*) é uma componente importante de muitas aplicações de processamento de linguagem natural(NLP), incluindo tradução automática, correção ortográfica, reconhecimento automático da fala, recuperação de informação, resposta a perguntas e ainda análise semântica [Yu+18]. *Word embedding* é o nome para um conjunto de técnias de aprendizagem de línguas em NLP e de modelagem de linguagem onde palavras ou frases do vocabulário são mapeadas para vetores. Existem muitos métodos diferentes para aprender *word embedding* a partir de um *corpus*, muitas vezes começando com uma codificação de uma só palavra, que mapeia cada palavra num vocabulário, composto por V elementos únicos, até um índice único num vetor. Representa-se então palavras através de vetores, vetores compostos tudo por zeros exceto por um 1 no digito adequado. Este tipo de representação é bastante atraente e intuitiva e permite que sejam efetuada operações matemáticas sobre esses vetores e ainda que sejam utilizados em algoritmos e estratégias de *Machine Learning*(ML). O Vector Space Model (VSM), geralmente contribuído a Salton (1975) e decorrente da comunidade de I*nformation Retrieval* (IR), é indiscutivelmente o modelo mais bem-sucedido e influente para codificar palavras e documentos como vetores. Uma boa representação de palavras deve ter algumas propriedades. Um avaliador de palavras ideal deve ser capaz de analisar modelos de *word embedding* a partir de diferentes perspetivas. A modelagem estatística da linguagem é um componente importante de muitas aplicações de NLP. Existem duas principais categorias de métodos de avaliação, os avaliadores intrínsecos e extrínsecos. Os avaliadores extrínsecos usam *word embedding* como *input features* para uma tarefa e medir as alterações de desempenho métricas específicas para essa tarefa. Alguns exemplos deste tipo de avaliadores são análise de sentimentos [RR15] e NER . Os avaliadores intrínsecos testam a qualidade de uma representação independente das tarefas de NLP, estes avaliadores medem diretamente as relações sintáticas ou semânticas entre as palavras. Como resultado do crescimento significativo de *word embedding*, muitos métodos forem propostos, métodos como *Neural Network Languague Model*, *Continuous-Bag-of-Word*, *Co-occurrence matrix*, *FastText*, *N-gram model*, *Dictionary model* e *Deep contextualized model*, *Word2vec*, *GlobalVectors*(GloVe) entre outros. Sendo que apenas serão abordados alguns destes métodos.

2.2.4 *N-grams*

Um *n-gram* é uma sequência de n *tokens* de bytes, caracteres ou de palavras [JM]. O conceito de *n-gram* foi introduzido pela primeira vez por Shannon sobre a Teoria da Informação, em 1948. Shannon usou *n-grams* baseados em caracteres para analisar e prever impressos

em inglês. Esta abordagem desde então tem sido aplicada a outras áreas como ortografia e correção de erros, compreensão de texto, identificação de linguagem , busca e recuperação de texto. O modelo *"bag-of-words"* não capta o verdadeiro significado semântico de uma palavra num determinado documento. A semântica é melhor capturada pela proximidade das palavras e sua ocorrência no documento [Oga16]. O fascínio pelos modelos de *n-gram* surge da sua simplicidade e interoperabilidade, nestes modelos o *corpus* de documentos é representado pela sua matriz de carateristicas *n-gram* em que cada linha e cada coluna, respetivamente, corresponde a um documento e *n-gram* distintos contando o número de ocorrências desse *n-gram* no documento [Pas17]. Jurafsky, Martin, Peter e Stuat em 2014 definiram um *n-gram* como uma sequência de n palavras: um *1-gram* (unigrama), um *2-gram* (ou *bigram*) é uma sequência de duas palavras, e um *3-gram* (ou trigrama) é uma sequência de três palavras. Cada um dos *n-gram* é uma coordenada de um vetor que representa o texto em estudo, e a frequência com que este *n-gram* aparece no texto pode ser também uma coordenada desse vetor. Um *token* é gerado por mover uma janela deslizante (*sliding window* sobre o *corpus* do texto onde o tamanho da janela depende do tamanho do token N e o seu deslocamento é feito em várias etapas. Cada etapa corresponde ou a um caracter ou uma palavra. Com base neste tipo de deslocamentos, os *n-gram* podem ser classificados como baseados em caracteres e baseados em palavras [MBR10]. O modelo de *n-gram* tem sido usado em muitas tarefas de NLP. Um método é o *ngram2vec*, que incorpora o modelo de *n-gram* em vários modelos de base de incorporação, tais como *word2vec*, *GloVe*, entre outros. Este tipo de modelos são usados também em modelos probabilisticos de *n-gram* para prever a próxima palavra após uma sequência de n-1 palavras. Por exemplo a chance da palavra *"disgusting"* se suceda a *Brussels sprouts* pode ser aproximada por dividir o numbero de vezes que o *4-gram Brussels sprouts are disgusting* aparece no texto em análise, pelo numero de vezes que o *(n-1)-gram Brussel sprouts are* acontece [Bar12]. A abordagem Goldilocks é uma abordagem que consiste em abordar através de uma *sliding window* de duas palavras, a palavra anteriror e a palavra seguinte. Estas abordagens também permitem que seja detetado negatividade no texto como por exemplo *"the staff was not friendly"*, onde a palavra *"not"* cancelam o valor positivo da palavra *"friendly"*.

2.2.5 *Stop-words*

Em computação, stop words são palavras que são filtradas antes do processamento de linguagem natural (NLP) dos dados, sendo geralmente as palavras mais comuns em cada idioma [Raj11]. Muitas vezes, existem palavras que podem ser consideradas omnipresentes, mas que são pouco úteis para alcançar a finalidade da análise, mas que aumentam ainda a dimensão do conjunto de recursos. Alguns exemplos destas palavras que podem ser consideradas irrelevantes podem ser: “as”, “e” , “os”, “de”, “sem”, “para”, entre outras.

Estas stop words são excluídas do vocabulário por completo como parte do processo de stop words removal. Esta exclusão é, por norma, motivada devido a dois fatores distintos:

1. Dimensão: remover stop words permite também reduzir de uma forma significativa os tokens existentes nos documentos, e desta forma, reduzir também a dimensão de recursos;

2. Irrelevância: Stop words são também muitas vezes palavras consideradas "vazias", pois geralmente não apresentam muito significado, o que traz ruído em processos de análise e/ou modelação. Devido a isto é possível uma análise apenas em palavras que realmente tenham conteúdo associado.

A conversão de todos os carateres em letras minúsculas antes do processo de stop word removal pode introduzir ambiguidade no texto e em alguns casos alterar por completo o seu significado. Exemplos disso podem ser expressões como "US citizen" que passa a ser visto como "us citizen" o que altera por completo o significado. Existem ainda outros exemplos que transformer estas stop words para minúsculas altera completamente o significado do texto e iria trazer resultados imprecisos. Existem vários esquemas que auxiliam neste tipo de casos, permitindo obter melhores resultados. O tf-idf pode ser usado com sucesso para filtragem de stop-words em vários campos de assunto, incluindo resumo de texto e classificação.

2.2.6 *TF-IDF*

Embora a TF-IDF seja uma pesagem relativamente antiga é simples e eficaz, tornando-o um esquema que serve como ponto de partida para outros algoritmos mais recentes [Buc88]. Essencialmente TF-IDF significa frequência de documentos invertidos, e o peso TF-IDF é um peso frequentemente utilizado em *information retrieval* e *Text Mining*. Este esquema pode ser categorizado como uma medida estatística, embora os seus resultados imediatos sejam determinísticos. A sua finalidade é determinar a relevância de uma palavra num documento ou num conjunto de documentos. Para isso utilizada a frequência relativa de palavras num determinado documento em comparação com a proporção inversa dessa palavra sobre um conjunto de documentos ou *corpus*. A importância aumenta proporcionalmente ao número de vezes que uma palavra aparece no documento, mas é compensada pela frequência da mesma palavra no *corpus*. Palavras que são comuns num único documento ou num pequeno grupo de documentos tendem a ter um valor de TF-IDF mais alto que palavras que estão presentes em todos os documentos, como preposições. Pode ser usado ainda com sucesso para filtragem de *stop-words* em vários campos como resumo de texto e classificação. No entanto, se uma palavra aparece muitas vezes num documento e não aparece muitas vezes em outros documentos, provavelmente significa que essa palavra é

importante e também lhe está associado um valor de TF-IDF maior. A implementação de TF-IDF tem algumas diferenças em todas as suas aplicações, mas têm todos a mesma forma base de funcionamento. O TF-IDF para uma palavra num documento é calculado através de multiplicar duas métricas diferentes. A primeira métrica é *term frequency*, ou a frequência do termo, que é simplesmente o número de vezes que uma palavra aparece em um documento. A segunda e última métrica é *inverse document frequency*, ou a frequência inversa do documento. O que por outras palavras seria o quão comum ou rara é uma palavra em todo o conjunto de documentos. Quanto mais próximo o seu valor for de o mais comum é esta palavra. Pode ser calculada como:

$$w_d = f_{(w,d)} * \log\left(\frac{|D|}{f_{(w,D)}}\right) \tag{1}$$

onde temos presente as duas funções e $f_{(w,d)}$ corresponde ao número de vezes que a palavra w aparece no documento d, $|D|$ é o tamanho do corpus e $f_{(w,D)}$ é igual ao número de documentos em que aparece em $|D|$ [Buc88]. Existem ainda várias situações diferentes que podem ocorrer para cada palavra, dependendo dos valores de $f_{(w,d)}$, $|D|$ e $f_{(w,D)}$. Como já foi referido este esquema pode ter várias aplicações, sendo uma delas information retrieval. O TF-IDF surgiu com o intuito de auxiliar nas pesquisas, para fornecer resultados que são mais relevantes para o que é procurado. Quando alguém pesquisa a palavra "Tesla" os resultados vão ser exibidos por ordem de relevância. Isso quer dizer que muitos dos artigos e noticias mais relevantes que contêm essa palavra serão classificados por ordem superior porque a TF-IDF dá a palavra "Tesla" uma pontuação mais elevada e dessa forma tornar-se mais relevantes sobre outros artigos que têm um valor de TF-IDF mais baixo ou não abordam esse assunto. Outra aplicação é keyword extraction, e neste tema o uso de TF-IDF também se tornou muito útil pois, uma vez que as palavras com maior valor de TF-IDF são as mais relevantes nesse documento, na maioria dos casos, podemos assumir que se tratam de palavras-chave nesse mesmo documento.

2.3 *keyword extraction*

A extracção de *keywords* é um problema importante no processamento de linguagem natural (NLP), onde o objetivo é identificar automaticamente num texto um conjunto de termos que melhor descreve o documento. As *keywords* podem servir como um pequeno resumo do documento, dando aos utilizadores uma visão geral do seu conteúdo, ou podem ainda indicar os tópicos que estão a ser discutidos. A abordagem mais fácil é talvez usar a frequência de uma palavra como critério para perceber se é relevante ou não e definir o quão importante essa palavra é no documento. Mas este tipo de abordagem oferece resultados muito pouco precisos. Desse modo outros métodos foram abordados e em

1999, Turney sugeriu um método onde regras heurísticas parametrizadas são combinadas com um algoritmo genético num sistema de extração de *keyphrases* (GenEx) que identifica automaticamente as *keywords* num documento [MT04]. Existem diferentes abordagens pra extração automática de *keywords*, tais como:

1. Abordagem Estatística: esta abordagem conta diretamente a frequência das palavras e é fácil de implementar, não necessita de treino nem de dados nem de carateristicas linguísticas. Pelo outro lado é impossível calcular e distinguir palavras e frases importantes em múltiplos documentos;

2. Abordagem Linguística: onde os nomes são considerados porque contêm uma grande quantidade de informação. As frases dos nomes são extraídas e recebem uma pontuação após a sua análise morfológica;

3. Abordagem *Machine Learning*: nesta abordagem a máquina é treinada de modo a que consiga encontrar *keywords* num documento e requer *high quality data*. Estes dados são divididos em 3 conjuntos: *train set*, *validation set* e *test set*. Um exemplo é *Keyphrase Extraction Algorithm* (KEA);

4. Outras Abordagens: em que combinam os conhecimentos dos outros métodos, como a posição da palavra, comprimento da palavra, tags html em torno das palavras, etc.

Um outro método bastante utilizado é o método TF-IDF, que foi referido anteriormente.

2.3.1 *Stemming*

A aplicação de um algoritmo stemming permite reduzir palavras à sua raiz. Este algoritmo funciona cortando o fim ou o início de uma palavra, tendo em conta uma lista de prefixos e sufixos comuns que podem ser encontrados em diferentes variações da mesma palavra. Este algoritmo ao ser aplicado permite, por exemplo, reduzir a palavra *sended* para *send*, ou *working* para *work*. Efetuar este corte pode ser bem-sucedido em algumas ocasiões, mas nem sempre o é, e por isso podemos afirmar que este algoritmo apresenta algumas limitações. Por norma, a aplicação destes algoritmos acaba por trazer uma diminuição do número de palavras no vocabulário e muitas das vezes as palavras que foram reduzidas podem não estar corretamente representadas, podendo até não serem compreensíveis do ponto de vista humano.

Num estudo realizado por Biba & Gjati em 2014, observaram que a aplicação destas técnicas de *stemming* num caso de classificação de texto melhorou os resultados obtidos quando comparados com os resultados obtidos sem *stemming*. Mas nem sempre a aplicação de *stemming* tem um impacto positivo nos resultados obtidos. Num estudo realizado por

Toman, Tesar, & Jezek em 2006, a aplicação destas técnicas causou um impacto negativo nos resultados obtidos, num caso específico também de classificação de texto.

Table 2.1: Exemplo de *Stemming*

Form	**Suffix**	**Stem**
Sended	-ed	Send
Working	-ing	Work
Niñas	-as	niñ

Esta redução de uma palavra para a forma raiz da palavra pode ser feita através de diferentes formas, como a remoção de sufixo e/ou prefixos de palavras, ou ainda a utilização de dicionários, onde é definida a raiz de várias palavras que podem ser ainda de outras línguas. Dentro destas diferentes formas destacam-se os algoritmos de *Porter Stemming*, *Snowball Stemming* e *Lancaster Stemming*. Nos anos 80 Porter propôs um algoritmo em que a sua principal preocupação era efetuar a remoção dos finais comuns das palavras, ou sufixos, para que elas voltassem a sua forma mais comum. Não é um algoritmo muito complexo, mas é dos mais antigos. Nos dias de hoje é visto apenas como um começo básico, mas não é aconselhável a ser utilizado. Este algoritmo foi posteriormente atualizado pois inicialmente apenas abrangia palavras em inglês e o autor descreveu em 2001 uma *framework* para criação de algoritmos *stemming* para que fosse possível a sua utilização noutras línguas [Poro1]. Um outro algoritmo de *stemming* é o *Snowball Stemmer*, também conhecido como Porter2 *stemmer*, uma vez que é um algoritmo mais recente e melhorado. Não é um algoritmo tão suave como o Porter stemmer pois foram adicionados vários aspetos ao *Snowball stemmer* devido a problemas no *Porter stemmer*. Por fim temos o algoritmo de *stemming* mais agressivo dos três que é o *Lancaster stemmer*. Este algoritmo pode ser usado no NLTK (*Natural Language Toolkit*) e permite que sejam adicionadas regras personalizadas ao mesmo. Este algoritmo como podemos dizer que é mais agressivo ocorrem falhas e pode tornar palavras quase impercetíveis para o leitor, ao contrário do *Porter* e *Snowball stemmer* que é mais intuitivo. É o algoritmo mais rápido e mais agressivo dos três, mas como reduz ainda mais o vocabulário não é recomendável de ser utilizado quando se procura ter alguma distinção

2.3.2 *Lemmatization*

O algoritmo de *lematization*, por outro lado, de modo a extrair o *lemma* próprio de cada palavra é necessário ter em consideração a análise morfológica das palavras. O *lemma* de cada palavra é simplesmente o seu dicionário ou forma canônica. Por exemplo o *lemma* de aviões é avião, e o *lemma* de cão é a palavra cão na mesma. Para isso, é necessário ter

dicionários detalhados que o algoritmo possa analisar para ligar a forma ao seu *lemma*. A chave para esta metodologia é linguística, pois para a *lemmatization* transformar uma palavra para o seu *lemma* ela precisa de conhecer a sua parte da fala. E isto requer um poder linguístico computacional extra, mas também traz outro tipo de vantagens que o algoritmo anterior não oferece, como transformar as palavras "é" e "será" para a sua forma canônica "ser".

Table 2.2: Exemplo de *Lemmatization*

Form	**Lemma**
Sended	Send
Working	Work
Niñas	niño

É de salientar que existe enquanto um lemma é a forma base de todas as suas formas inflecionais, uma stem não é. Por esse motivo, os dicionários regularmente são listas de lemas e não de stems. Este fato pode gerar duas consequências distintintas:

1. Uma das consequências possíveis é que a mesma stem tenha diferentes *lemmas*, o que afeta os resultados da pesquisa;
2. E é possível ainda que o mesmo Lemma corresponda a várias formas de *Stem*, e é necessário tratar essas formas como a mesma palavra.

2.4 *text summarization*

Nos últimos anos, tem havido um grande aumento da quantidade de dados em forma de texto proveniente de variadas fontes. Esta informação toda é um recurso cada vez mais importante e apesar de existir cada vez mais um volume inestimável de informação e conhecimento, este precisa de ser eficazmente resumido para ser útil. Esta quantidade crescente de informação exigiu que houvesse uma pesquisa exaustiva na área de *text summarization*. Um resumo de um texto é definido segundo Eduard Hovy como um texto que provém de um ou mais textos e contém informação importante do texto original mas que tem metade do comprimento do texto originial ou geralmente é ainda mais curto [Buso5]. Devio ao facto que é difícil para seres humanos resumir um documento seria impensável fazê-lo para uma grande quantidade de documentos. Desse modo surge *automated text summarization*, que é o processo de gerar automaticamente resumos para um determinado texto enquanto é preservada informação crucial [Erco6]. *Automatic text summarization* surgiu já nos anos 50, surgiu um método que extraía frases de um texto através da frequência de

palavras e frases. Na altura propuseram ponderar as frases de um documento em função de palavras de alta frequência ignorando palavras de frequência muito alta. Desta maneira a cada palavra do documento era atribuído um valor baseado na sua frequência, por exemplo se a palavra "computador" aparecesse 9 vezes era associado o valor 9 a essa palavra. Desse modo posteriormente era associado um *score* para essa frase, sendo este a soma desses valores. Mas desse modo frases maiores iriam por norma ter um *score* superior, e por isso finalmente era dividido esse *score* pelo comprimento de cada frase [Luh58]. Harold P Edmundson, em 1969 descreveram o paradigma com base em *keywords* que, além da frequência padrão dependem também dos "pesos". Decidiram então utilizar três métodos para determinar o "peso da frase":

1. Método *Cue*: A relevância de uma frase é calculada com base na presença ou ausência de certas *keywords* no dicionário Cue;

2. Método *Key*: È baseado na hipótese de que as *keywords* com alta frequência no documento são relevantes.

3. Método do Título: O peso de uma frase é calculado como a soma de todas as palavras com conteúdo que aparecem no título e nos sub-títulos de um documento;

4. Método de Localização: Este método é baseado no suposto que: as frases que ocorrem sob certas cabeçalhos são positivamente relevantes; e as frases relevantes tendem a ocorrer muito cedo ou muito tarde em um documento e seus parágrafos.

Os resultados mostram que a melhor correlação entre as extrações de origem humana e de origem automática foram obtidos pela combinação destes quatro métodos. Existem duas formas principais de *text summarization*:

1. Extrativo: que é um método para encontrar algoritmicamente as frases mais informativas dentro de um grande corpo de texto que são usadas para formar um resumo. Este método depende assim apenas da extração de frases do texto original;

2. Abstrativo: que é um método para gerar algoritmicamente frases concisas que são semanticamente consistentes com corpo do texto. Este método é mais próximo do método que os seres humanos utilizam para resumir o texto, uma vez que utiliza técnicas de NLP, mas também é mais difícil de implementar.

Resumos puramente extrativos frequentemente vezes dão melhores resultados em comparação com resumos de abstração automáticos. Como todo os métodos existem sempre problemas associados. No caso do resumo do tipo extrativo alguns dos problemas são:

- As frases extraídas tendem geralmente a ser mais longas do que a média;

- As informações importantes ou relevantes estão geralmente espalhadas pelas frases, e os resumos extrativos não conseguem capturar isso;
- Informações contraditórias não podem ser apresentadas com precisão.
- A extração pura conduz frequentemente a problemas em geral. As sentenças contêm frequentemente pronomes, que perdem as suas referências quando extraídas fora do contexto e pode levar a uma interpretação que não a correta.

Falando agora dos métodos abstrativos, o maior desafio para o resumo abstrativo é o problema da representação. As capacidades dos sistemas são a sua capacidade de gerar tais estruturas (os sistemas não podem resumir o que suas representações não podem capturar). limitadas pela riqueza de suas representações
MELHORAR ISTOOOOOOOOOOOOOOOOOOOOOOOOOO

2.5 NER

Named Entity Recognition ou NER é um processo de information extraction que tem como finalidade através de uma frase ou um pedaço de texto, analisá-lo para encontrar entidades que podem ser classificadas em categorias predefinidas, tais como nomes de pessoas, organizações, lugares, quantidades, valores monetários, percentagens e muitos mais exemplos.

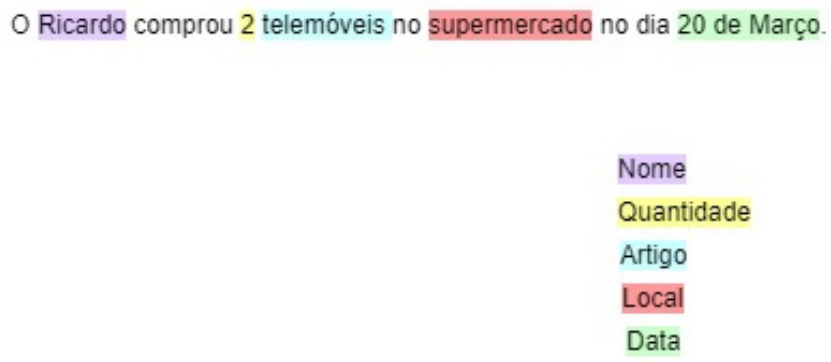


Figure 2.1: Exemplo de frase onde foram classificadas entidades

O conceito de *Named Entity Recognition*, numa primeira instância, foi abordado na área de extração de informação, na sexta edição da *Message Understanding Coference* [GS96], no entanto, tem sido uma tarefa bastante útil na área de *Text Mining*. Foi ainda considerada uma parte importante da aplicação de *Text Mining* com o propósito de analisar literatura, nomeadamente na área da biomedicina [Tan+05]. Esta tarefa pode ter duas diferentes abordagens, a primeira baseando-se em regras e a segunda uma abordagem que utiliza *Machine Learning* recorrendo a métodos estatísticos. Uma vez que o conceito de NER é

considerado uma função básica de NLP é desafiado pelas várias complexidades inerentes a qualquer linguagem natural. Um dos principais desafios é a linguagem. O reconhecimento de palavras que podem ter múltiplos significados ou palavras que podem fazer parte de frases diferentes. Outro desafio e talvez o maior é classificar palavras semelhantes, múltiplas palavras podem ser escritas de diferentes formas, podem ser abreviadas para facilitar a escrita e a sua compreensão. Palavras que não são usadas com muita frequência hoje em dia é outro grande desafio e ainda palavras que estejam noutra língua. Existem diferentes modelos que têm vindo a ser aplicados, modelos como árvores de decisão, *Support Vector Machines*, *Neural Networks* de várias arquiteturas [Lam+16]. Como a linguagem é um dos principais obstáculos para a aplicação de NER, um modelo proposto por Nothman, Ringland, Radford, Murphy, & Curran em 2013 explorou o facto de a Wikipedia conter grande quantidade de dados, e em várias línguas para a criação de dados de treino e treinar modelos capazes de identificar entidades em diferentes idiomas.

Dicionários de entidades podem ser utilizados na abordagem baseada em regras. Nesses dicionários são definidos os mapeamentos entre termos e a respetiva classificação, como por exemplo, o dicionário pode ter o mapeamento da palavra "Ricardo" e a classificação "Nome". Por fim, nas abordagens que utilizam *Machine Learning* é possível combinar dados previamente classificados com dados que ainda não tenham sido classificados. Isto é uma solução possível para casos em que é necessário treinar um modelo, com um número de dados previamente classificados é reduzido [LV09]. Esta abordagem pode-se dizer semi-supervisionada, mas ainda existem outras como a proposta por [ZE13], em que não exige que os dados estejam previamente classificados como dados de treino, que já é considerada uma abordagem supervisionada

2.6 *topic modeling*

A compreensão de grandes coleções de textos continua a ser um problema. *Topic modeling* oferece um formalismo para expor os temas de uma coleção e têm sido usados para ajudar em IR (*information retrieval*) [WC06] e entender ideias científicas [Hal+10]. Em 1998, Papadimitriou, Raghavan, Tamaki e Vempala descreveram um modelo temático inicial, chamado probabilistic *latent semantic analysis* (PLSA). Este método, *latent semantic analysis*, é o mais básico e analisa a frequência das palavras dentro de um documento e cria tópicos baseados nas frequências de palavras que ocorrem em cada documento [Ste+04]. Outro modelo que foi criado por Thomas Hofmann em 1999 chamado de *Latent Dirichlet allocation* (LDA) é talvez o *topic modeling* mais comum atualmente em uso e é uma generalização do PLSA. Estes *topic models* mostrados pelo LDA são atraentes porque descobrem grupos de palavras que muitas vezes aparecem juntas em documentos. Estes são os "tópicos" dos *topic models* e são distribuições multinominais sobre as palavras do vocabulário. As palavras que

têm a mais alta probabilidade num tópico revelam do que se trata esse tópico [BLo6]. A vantagem de utilizar *topic models* é que estes modelos não são supervisionados. Eles não requerem nenhuma anotação "*a priori*". A única entrada que um *topic model* requer é o texto dividido em documentos e o número de tópicos que se pretende obter. *Topic modeling* é uma ferramenta em *Text Mining* frequentemente utilizada para a descoberta de estruturas semânticas ocultas num corpo de texto. Intuitivamente, dado que um documento é sobre um tópico em particular é expectável que determinadas palavras apareçam no documento com mais ou menos frequência. Por exemplo um documento relacionado sobre tecnologia tem grande probabilidade de aparecerem palavras como "computador, sistema, telemóvel, *internet*" e num documento sobre desporto palavras como "jogador, vitória, campeão, bola" aparecerão mais frequentemente. Enquanto que palavras como "o", "a" e "que" aparecerão igualmente em ambos. É possível consideram um tópico como "bom" quando tem coerência semântica, ou seja, que os tópicos tenham significado suficiente para que os utilizadores compreendam o tema principal. Um problema fundamental é que mesmo que tudo corresse perfeitamente, a função objetiva que os *topic models* otimizam nem sempre se correlaciona com os julgamentos humanos da qualidade do tema [TR13]. O que significa que nem sempre os *topic models* fazem sentido para os utilizadores finais. O processo de geração do modelo LDA tem um para cada documento d em um corpus D, assume que:

1. Escolhe-se a *topic distribution* θd a partir de um Dirichlet anterior uniforme com o parâmetro α.
   - $\theta d \sim Dir(\alpha)$
2. Para cada palavra $W_{(d)}$ no documento d:
   a) Escolhe-se um tópico $z_{(d)}$,d de uma distribuição multinomial com o parâmetro θd.
      - $z_{(w,d)} \sim Multinomial(\theta d)$
   b) Escolhe-se uma palavra $W_{(d)}$ de uma probabilidade multinomial com o parâmetro β condicionado sobre o tópico $z_{(w,d)}$.
      - $w_{(d)} \sim Multinomial \beta z_{w,d}$

2.7 ANÁLISE DE SENTIMENTO

Sentimental analysis, ou análise de sentimento, é o campo que analisa as opiniões, sentimentos, avaliações, atitudes e emoções das pessoas em relação a algumas entidades, como produtos, serviços, organizações, indivíduos, tópicos, etc. Os termos análise de sentimento, mineração de opinião, extração de opinião, mineração de sentimentos, análise de subjetividade, análise de efeitos, análise de emoções, mineração de revisão pertencem à mesma categoria comumente conhecida como análise de sentimentos [Liu12a]. Sentimental analysis

muitas vezes é conduzida em três níveis conhecidos como Nível de Documento, Nível das Frases e Nível de Entidade e Aspeto. A análise do sentimento no Nível do Documento classifica o documento inteiro em positivo ou negativo [PLV02]. A classificação do Nível de Frases classifica a frase na categoria positiva, negativa ou neutra. Nível de entidade e nível de aspeto também conhecido como sentimental analysis de nível de característica dá o resumo sobre qual característica de um produto o utilizador gosta ou não gosta [NS12].

2.7.1 *Machine Learning*

Mitchell em 1997 definiu "*machine learning*" como: um programa de computador é dito para aprender com a experiência E e com respeito a alguma classe de tarefas T e como medida de desempenho P nas tarefas T, que melhora com a experiência E. A medida de desempenho P quantifica o desempenho de um determinado algoritmo. Os algoritmos de aprendizagem são submetidos a um conjunto de dado, ou experiência E. Esses conjuntos de dados por norma contêm um conjunto de exemplos que são usados para treinar e testar esses algoritmos [JFM96]. *Machine Learning* é a tarefa de programar computadores para otimizar um critério de desempenho utilizando dados de treino ou experiência passada [Adm13]. O estudo e a modelação computadorizada dos processos de aprendizagem em suas múltiplas manifestações constituem o tema de *Machine Learning* [TR13]. *Machine leraning* é um método de análise de dados que automatiza a construção de modelos analíticos, que baseia na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana. O primeiro programa capaz de *self-learning* foi criado pelo cientista A. Samuel em 1952, este programa tornou-se o melhor no jogo de damas [Bro10]. Desde os anos 90, *machine learning* é usado em diversas áreas de pesquisa, tais como: *Text Mining* , NLP, reconhecimento facial [TR13], deteção automática de sinais de trânsito , *speech recognition* [Rab89], *image recognition*, análise de sentimento entre outros.

2.7.2 *Supervised Learning*

Machine learning pode ser feito tanto de uma forma supervisionada como não supervisionada. Quando se fala de *supervised learning*, fala-se de um sistema que recebe um conjunto de dados com diferentes exemplos de valores de parâmetros e decisões. Dos quais infere uma função matemática que vai mapear variaveis de *input*(x) para variaveis *output*(y), onde o objetivo principal é aproximar a função de mapeamento tão bem, de modo a quando seja introduzidos novos *inputs*(x) seja possível prever os *outputs*(y). A sua forma mais básica pode ser escrita como:

$$Y = f(X) \tag{2}$$

Este tipo de aprendizagem deduz uma função chamada de *training data* que consite num conjuto de vários exemplos de treino. O nome de *supervised learning* provém do modo como o processo funciona. O processo de treinar um *dataset* é muitas vezes associado a um professor a supervisionar todo o processo de aprendizagem, uma vez que, o algoritmo faz previsões iterativas sobre os dados de treino e é corrigido pelo professor. Esta aprendizagem termina quando o algoritmo atingir um nível de desempenho aceitável. Numa outra abordagem, *unsupervised learning*, significa que o sistema atua e observa as consequências dos seus atos, sem se referir a nenhum caso pré-definido que não seja aquele observado anteriormente. Este método é caraterizado pela sua "aprendizagem pela prática", ou tentativa e erro. Nesta aprendizagem so existem os dados de *input*(x) e nenhuma variável de *output*(y) correspondente. O objetivo desta aprendizagem é de modelar a estrutura ou distribuição subjacente nos dados a fim de aprender mais sobre os mesmos. Este tipo de métodos numa fase mais inicial apresentam mau desempenho mas à medida que vão "aprendendo" o seu desempenho aumenta. Uma tarefa de *supervised learning* é chamada de regressão quando os *outputs*(y) são valores numéricos e chamada de classificação quando os *outputs*(y) tomam valores categóricos ou discretos. Por exemplo quando se tenta prever um preço de um telemóvel a partir de um *dataset* é um caso de regressão pois o preço vai ser um *Output* continúo. Alguns algoritmos mais comuns de regressão são *linear Regression, Support Vector Regression(SVR)* e *Regression Trees*. Num mesmo caso dado um mesmo *dataset* sobre telemóveis, seria um problema de classificação quando o objetivo do algoritmo fosse prever se o preço dos telemóveis vai ser superior ou inferior ao preço de retalho, ou seja, o *output* vai ser sempre dentro de duas categorias, superior ao preço de retalho, ou inferior ao preço de retalho. Alguns exemplos comuns de algoritmnos de classificação são *Naive Bayes, Decision trees, K Nearest Neighbors* e *Logistic Regression*.

2.7.3 *Modelos ML*

Existem diferentes modelos de *Machine Learning* e cada um serve um propósito diferente. Desse modo, não é viável pré-definir os algoritmos de *Machine Learning* que se adequam melhor ao intuito deste projeto. Por isso vão ser enunciados alguns que são bastante comuns: *Decision Trees; Naive Bayes; SVM; Random Forest* e kNN.

Descision Trees

As árvores de decisão, ou *decision trees*, são um dos métodos de aprendizagem mais utilizados e têm sido aplicados com sucesso a um grande número de tarefas [Mit97]. Novas instâncias são classificadas ordenando-as pela árvore a partir do nó raiz de acordo com os valores dos atributos testados em nós sucessivos. Cada ramo que descende de um nó corresponde a um dos valores possíveis para este atributo. O processo continua até atingir um nó de

folha que forneça a classificação da instância [Mit97], [WFH]. Para construir uma árvore de decisão, um atributo é selecionado para colocar no nó raiz e é criado um ramo para cada valor possível do atributo. Isso divide o exemplo definido em subconjuntos, um para cada valor do atributo. O processo pode ser repetido recursivamente para cada ramo utilizando apenas aquelas instâncias que realmente alcançam o ramo. Se a qualquer momento todas as instâncias de um nó tiverem a mesma classificação, não é necessário desenvolver mais essa parte da árvore [WFH].

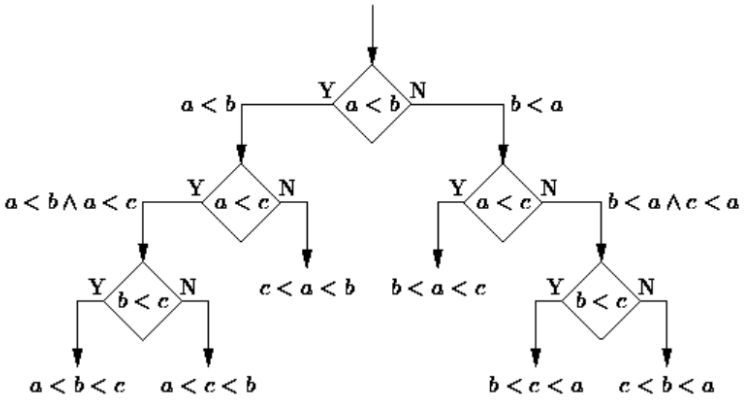


Figure 2.2: Gráfico de uma *Decision tree*, Bruno R. Preiss, P.Eng. Al, 1997

Fundamental para o desenvolvimento de qualquer árvore de decisão é a seleção de uma forma ótima de dividir os dados, ou seja, selecionar o atributo mais útil para classificar os exemplos. [WFH]. O método de classificação por árvore de decisão destaca-se de outras ferramentas de apoio à decisão com várias vantagens, sendo a principal vantagem a sua simplicidade na compreensão e interpretação, mesmo para utilizadores não experientes. O decision tree learning é adequado para problemas onde as instâncias são representadas por pares de valores de atributos e a função alvo tem valores de saída discretos [Mit97]. Pode-se demonstrar experimentalmente que as tarefas de classificação de texto frequentemente envolvem um grande número de características relevantes [Joa98]. Portanto, a tendência de uma árvore de decisão de basear as classificações no menor número possível de testes pode levar a um mau desempenho na classificação do texto. Os [Kim+14] descrevem uma aplicação de árvores de decisão para personalização de anúncios em páginas *web*. O maior risco de implementar uma árvore de decisão é que ela se ajusta mais aos dados de treino com a ocorrência de uma árvore alternativa que categorize pior os dados de treino, mas que por outro lado categorize melhor os documentos a serem categorizados [GL+09].

Naive Bayes

O teorema de *Bayes* fornece uma forma de calcular a probabilidade de uma hipótese com base na sua probabilidade anterior. Suponha que existe uma classe H e alguns dados de

treino D observados. Então P(h) é a probabilidade de que a hipótese h seja verdadeira, ou seja, é a probabilidade anterior de h. P(d) é a probabilidade anterior dos dados de treino d. $P(D|h)$ é a probabilidade de observar os dados d, dado que a hipótese h se mantém. O teorema de *Bayes* fornece uma forma de calcular a probabilidade condicional $P(h|D)$ da probabilidade anterior P(h), juntamente com P(d) e $P(D|h)$ e é dado por:

$$P_{(h|D)} = \frac{P_{(D|h)}P(h)}{P(D)} \tag{3}$$

Para determinar a hipótese mais provável, dados os dados de treino D observados, é selecionada a hipotese *"maximum a posteriori"*(MAP). A MAP é determinada usando o teorema de *Bayes* para calcular a probabilidade *a posteriori* de cada hipótese e é dada por:

$$\begin{aligned} h_{MAP} &= argmax_{h\epsilon H} P(h|D), \\ &= argmax_{h\epsilon H} \frac{P(D|h)P(h)}{P(D)}, \\ &= argmax_{h\epsilon H} P(D|h)P(h). \end{aligned} \tag{4}$$

Na etapa anterior de P(D) é descartada por ser uma constante independente de h [Mit97], *Naive Bayes* é um modelo probabilístico não paramétrico baseado no teorema de *Bayes* [Mic68]. Neste teorema assume-se que o efeito de um valor de variável sobre uma determinada classe é independente dos valores de outras variáveis. Esta suposição é chamada de independência condicional entre os atributos das variáveis. A suposição de independência torna muito mais fácil estimar estas probabilidades já que cada atributo pode ser tratado separadamente, resultando na equação seguinte:

$$v_{nb} = argmax_{v_j \epsilon V} P(v_j) \prod_{i=1..k} P(u_i|v_j) \tag{5}$$

O número de termos $P(u_i|v_j)$ que devem ser estimados a partir dos dados de treino é igual ao número de valores de atributos multiplicado pelo número de valores-alvo, que é um número consideravelmente menor do que se a suposição de independência condicional não fosse feita [Mic68]. Embora esta suposição seja frequentemente violada, os classificadores *Bayes*, ingênuos, têm demonstrado funcionar surpreendentemente bem e, em alguns casos, realizam métodos mais sofisticados. Uma vantagem do classificador *Naive Bayes* é que ele requer uma pequena quantidade de dados de treino para estimar os parâmetros necessários para a classificação. A abordagem da classificação *Bayesiana* chega à classificação correta desde que a categoria correta seja mais provável que as outras. *Naive Bayes* trabalha bem com dados numéricos e textuais, fáceis de implementar e de calcular em comparação com outros algoritmos, porém a suposição de independência condicional é violada por dados

do mundo real e tem um desempenho muito fraco quando as características são altamente correlacionadas e não considera a frequência de ocorrências de palavras.

Support Vector Machine(SVM)

Support vector machines(SVM) é um dos métodos de classificação discriminatória que é geralmente reconhecido como sendo dos métodos mais precisos. O método de classificação SVM é baseado no princípio da *Structural Risk Minimization* a partir da teoria de *Machine Learning* [Joa98]. A ideia deste princípio é encontrar uma hipótese para garantir o menor erro verdadeiro. Além disso, os SVM são bem fundamentados e muito abertos ao entendimento teórico e à análise [PLV02]. O SVM precisa de um conjunto de treinos positivos e negativos, este tipo de requisito é incomum para outros métodos de classificação. Esses conjuntos de treino positivo e negativo são necessários para que o SVM procure a superfície de decisão que melhor separe os dados positivos dos negativos no espaço n-dimensional, o chamado hiperplano. A abordagem básica para classificar os dados, começa por tentar criar uma função que divide os pontos de dados com a menor quantidade possível de erros ou então com a maior margem possível. Isso se deve ao fato de que áreas vazias maiores ao lado da função de divisão resultam em menos erros. Os representantes do documento que estão mais próximos da superfície de decisão são chamados de vetor de suporte. O desempenho da classificação SVM permanece inalterado se os documentos que não pertencem aos vetores de suporte forem removidos do conjunto de dados de treino [BKM02]. Cada objeto é representado por um vetor. Assumindo que os dados devem ser divididos em duas categorias diferentes, o classificador de margem máxima tenta encaixar um hiperplano, ou limite de decisão, entre as duas classes que separam os objetos de treino no espaço de características. A margem entre as duas classes é maximizada para que os outros pontos sejam mais provavelmente ser classificado corretamente no grupo certo. No exemplo da imagem abaixo representa o hiperplano A que cabe numa área vazia maior entre as duas classes do que o hiperplano B, portanto, o hiperplano A também é chamado hiperplano de margem máxima. Neste caso, a separação A é a melhor, pois distingue as duas classes de uma forma mais precisa.

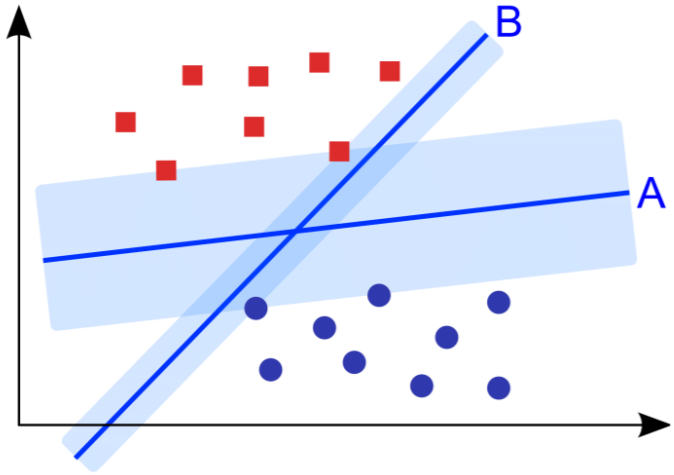


Figure 2.3: Duas linhas divisórias possíveis com tamanhos de margem diferentes. W. Commons, 2010

Muitas vezes não é possível separar linearmente os dados de treino, como na figura anterior, mas é mais provável que sejam distribuídos como nesta figura.

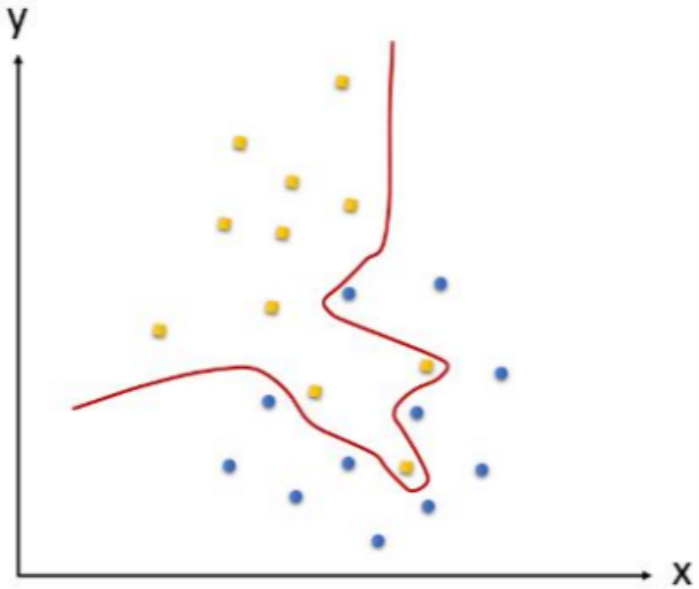


Figure 2.4: Objetos não separáveis linarmente, Tatjana Bürgmann, 2015

Isto pode ser causado devido a erros de medição ou apenas porque as duas categorias se sobrepõem naturalmente. Neste caso, o hiperplano não pode ser ajustado entre as duas classes e o classificador de margem máxima não pode ser usado para este tipo de dados. Com o objetivo de realizar uma classificação não linear, SVM usa o chamado *kernel trick*. O *kernel trick* converte o espaço vetorial e todos os vetores de treino num espaço de características de

dimensão superior, de modo a que os vetores de treino se tornem linearmente separáveis e o hiperplano possa caber entre eles. Ao transformar o hiperplano de volta para o espaço de característica inferior, o hiperplano deixa de ser linear, ou seja, é neste momento que o *kernel trick* "entra em jogo". Usando as funções do *kernel* para descrever o hiperplano no espaço de características de dimensão superior, o hiperplano resultante no espaço de características de dimensão inferior é ainda matematicamente descritível. Dessa forma é possível realizar a transformação para um espaço de características de dimensão superior e voltar sem ter de implementar realmente as transformações computacionalmente. *Support vector machines* é a melhor técnica para documentos de classificação [LKR09]. Contudo, a maior desvantagem da SVM é o seu treino relativamente complexo e os algoritmos de categorização e também o alto consumo de tempo e memória durante a fase de treino e classificação. Além disso, as confusões que ocorrem durante as tarefas de classificação devido aos documentos podem ser notadas em várias categorias devido à similaridade que normalmente é calculada individualmente para cada categoria [BKM02].

Random Forest

O método geral *Random Forest* ,ou *random decision forest*, foi proposto pela primeira vez por Ho em 1995 [Ho98]. Ho estabeleceu que florestas de árvores divididas por hiperplanos oblíquos podem ganhar precisão à medida que crescem sem sofrerem de excesso de treino, desde que as florestas sejam restritas aleatoriamente para serem sensíveis apenas a dimensões de características selecionadas. *Random forest*(RF) , é um método de aprendizagem em conjunto para classificação, regressão e outras tarefas que operam através da construção de uma multiplicidade de árvores de decisão no momento do treino e produzem a classe(classificação) ou previsão média (regressão) das árvores individuais [Ho98]. Porquê o nome *Random Forest*? Devido ao facto que por mais que as pessoas possam confiar em diferentes fontes para fazer uma previsão, cada árvore de decisão na "floresta" considera um subconjunto aleatório de características ao formar perguntas e só tem acesso a um conjunto aleatório dos pontos de dados de treino.

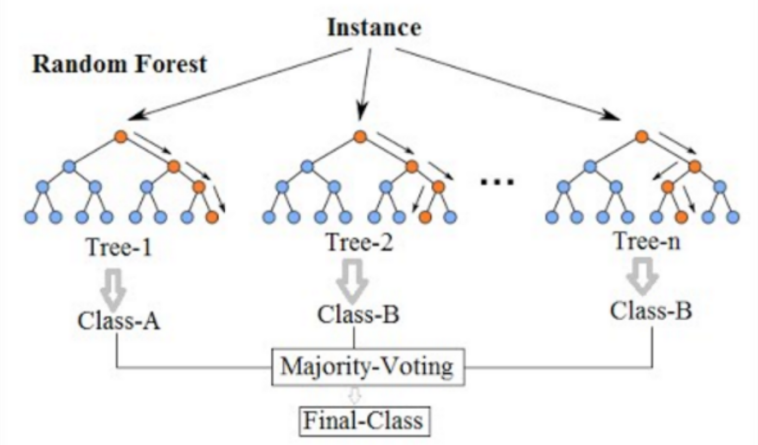


Figure 2.5: Exemplo de de uma Random Forest, Venkata Jagannath, 2017

Ao contrário das árvores de decisão , a RF é capaz de levar em conta mais de uma característica em um nó. Ao utilizar a recolha aleatória de amostras e a maioria dos *outputs* de várias árvores de decisão, a RF também tem a vantagem de ser insensível ao ruído nos dados ou ao excesso de treino. Além disso, em consequência da recolha aleatória de amostras, a RF é muito eficiente em grandes quantidades de dados, relativos número de aulas, amostras e características. Entretanto, o tempo de treino para uma *Random Forest* aumenta linearmente com o número de árvores.

k-Nearest Neighbor(kNN)

Este algoritmo, *k-nearest neighbour*(kNN), é usado para testar o grau de semelhança entre documentos e dados de treino k. Serve também para armazenar uma certa quantidade de dados de classificação, determinando assim a categoria dos documentos de teste. Este método é um algoritmo de aprendizagem instantânea que categoriza objetos com base no espaço de características mais próximas no conjunto de treino [HKK01]. O algoritmo *k-Nearest Neighbour*, ou em português "vizinho mais próximo", é um dos mais simples e populares de todos os algoritmos de *Machine Learning*. O algoritmo simplesmente compara um vetor de características a ser classificado com todas as características vetoriais do conjunto de treino com as etiquetas de classe conhecidas, e atribui ao vetor a classe mais frequente. Para encontrar o vizinho mais próximo, é utilizada uma medida de distância. A distância *Euclidean* é geralmente a métrica mais utilizada, mas outras métricas também podem ser usadas, como por exemplo a distância de *Mahalanobi*, a distância de *Manhattan* ou a distância de *Canberra*.[WFH].

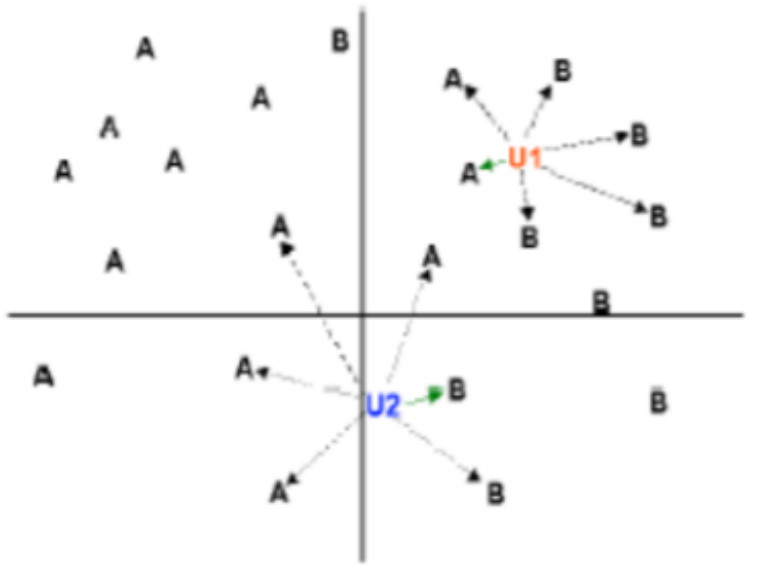


Figure 2.6: Exemplo de um gráfico do algoritmo *k-nearest neighbour*

A principal vantagem dos algoritmos kNN é que eles são facilmente compreendidos e implementados. Apesar de ser simples, o algoritmo apresenta um bom desempenho em muitos casos, especialmente para classes multimodais. Este tipo de classificação, embora muitas vezes com bons resultados, tem uma série de inconvenientes, como a necessidade de definir o número k(vizinhos), o que tem impacto direto no desempenho final do algoritmo. Para além disso, como cada exemplo de treino é considerado para a classificação kNN é necessário que sejam armazenados, o que afeta o desempenho e traz altos custos de armazenamento. Um método para melhorar este aspeto é a eliminação de amostras redundantes em regiões consideradas estáveis. Também é necessário ter em conta a necessidade de reduzir o cálculo e o tempo necessário para classificar um novo exemplo. Para esse efeito, técnicas como a *kd-tree* podem ser implementadas para indexar os exemplos de treino armazenados [Mit97]. Um outro aspeto que é relevante é que caso a vizinhança seja muito pequena não irá conter nenhum ponto de dados, mas se for demasiado grande irá conter todos os pontos de dados. Chawla e Liu em um de seus recentes trabalhos [LC11] apresentaram uma nova estratégia de ponderação *k-Nearest Neighbors* para lidar com o problema do desequilíbrio de classes. Eles propuseram CCW (*class confidence weights*) que usa a probabilidade de valores de atributo dados rótulos de classe para ponderar protótipos em kNN. Enquanto o kNN regular usa diretamente as probabilidades dos rótulos de classe na vizinhança da instância de consulta, eles usaram as probabilidades condicionais das classes. Eles também mostraram como calcular os pesos CCW usando a modelagem de misturas e redes *Bayesianas*. O método foi executado com mais precisão do que os algoritmos de estado da arte existentes.

2.7.4 *Métricas de avaliação*

Métricas são recursos relacionados especificamente a cada abordagem de *Machine Learning*. Existem diferentes métricas associadas a diferentes tipos de tarefas, como por exemplo: classificação, regressão, *ranking*, *clustering*, *topic modeling*, entre outros. Sendo que no âmbito deste projeto o foco será sobre classificação. Um exemplo clássico de classificação binária é o exemplo chamado de "detecção de *spam*", onde se tem como exemplo o conteúdo do *e-mail* e informações sobre o remetente, com a finalidade de classificar um *e-mail* como *spam* ou não *spam*. Após ter sido definido o problema e tratado e analisado os dados, procede-se à aplicação de modelos de *Machine Learning*. É necessário predefinir métricas para avaliar o desempenho dos modelos utilizados, existem diversas formas de avaliar um modelo, mas em segue a lista das principais métricas apontadas para avaliar os modelos descritos em cima:

Matriz de confusão

Esta matriz oferece 4 métricas distintas sobre o modelo em estudo (*true positive, true negative, false positive, false negative*). T*rue positive* (TP) indica a quantidade de registos que foram corretamente classificados como positivos, ou seja, a resposta do classificador foi que o comentário era positivo e o comentário realmente era positivo. *True negative* (TN) indica a quantidade de registos que foram corretamente classificados como negativos, ou seja, a resposta do classificador foi que o comentário era negativo e o comentário realmente era negativo. *False positive* (FP) indica a quantidade de registos que foram classificados incorretamente como comentários positivos, ou seja, a resposta do classificador foi que o comentário era positivo, mas o comentário era negativo. *False negative* (FN) indica a quantidade de registos que foram incorretamente classificados como comentários negativos, ou seja, a resposta do classificador foi que o comentário era negativo, mas o comentário era positivo. Através desses quatro valores, seremos capazes de calcular os indicadores: *Accuracy, Precision, Recall e F1 Score*.

- O cálculo da *accuracy* é bastante simples, visto que consiste simplesmente em avaliar quão frequentemente o modelo de classificação efetua uma previsão correta.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

- O cálcuo do indicador de precisão do modelo, *precision*, este cálculo serve principalmente para indicar a relação entre as previsões positivas realizadas corretamente e as previsões positivas totais, desse modo , serve para reduzir a existência de *false positives*.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

- O indicador *Recall*, ou também conhecido por *True Positive Rate*, é utilizado para identificar a razão entre os valores previstos como sendo positivos e todos os valores. Serve essencialmente para minimizar os false negatives.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

- Já o indicador *F1 Score*, já não é tão simples e intuitivo de entender como os indicadores a cima descritos, mas este indicador permite que seja visualizado as métricas de *precision* e *recall* juntas.

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

  Este indicador é considerado melhor que o de accuracy em muitos casos, principalmente em casos onde false positives e false negatives possuem um impacto diferente nos modelos.

Uma outra métrica que pode ser relevante para a avaliação das modelos é AUC (*Area Under the ROC Curve*). Esta métrica mede a área total sobre a curva de ROC. A curva de ROC é um gráfico bi-dimensional de coordenadas (*True Positive Rate, False Positive Rate*) que mostra o desempenho de um modelo de classificação.

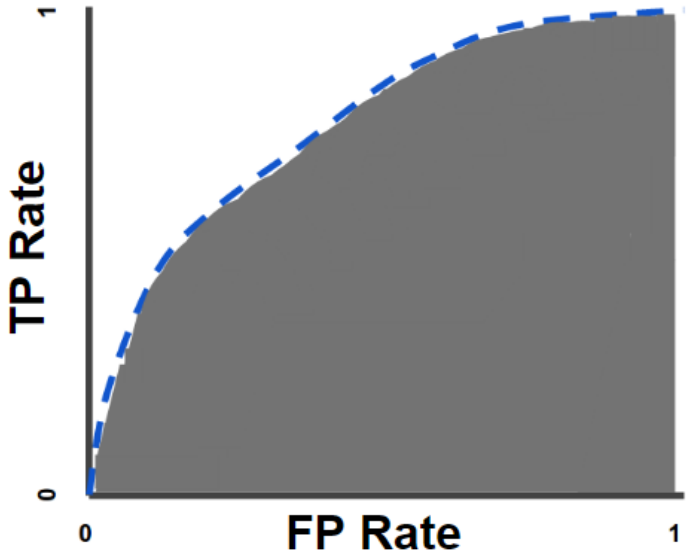


Figure 2.7: Exemplo de um gráfico AUC, adaptado de https://bit.ly/32Y11io

O objetive de utilizar esta métrica é lidar com situações em que a distribuição é bastante dispersa, e a curva "equilibra" os tamanhos das classes.

2.8 TECNOLOGIA

A nível tecnológico, para o desenvolvimento desta dissertação serão utilizadas algumas ferramentas, sendo algumas relacionados com a área de *Text Mining*, e alguns pacotes. Segue então a lista: *Python; Anaconda; PyCharm; R; RStudio; NLTK*.

- **Python**: Segundo (www.monkeylearn.com) *Python* é a linguagem mais utilizada na computação científica atualmente. Ferramentas como *NumPy* e *SciPy* a estabeleceram como uma linguagem rápida e dinâmica que chama bibliotecas C e Fortran onde o desempenho é necessário. *Phyton* oferece ainda uma comunidade que cresce diariamente e um conjunto diversificado de bibliotecas para implementar modelos de NLP. Estes aspetos tornaram a linguagem de *Phyton* uma das mais utilizadas para análise de texto.

- **PyCharm**: PyCharm é uma plataforma híbrida desenvolvida por *JetBrains* como uma IDE para Python. É usualmente utilizado para o desenvolvimento de aplicações Python, sendo dos IDEs mais utilizados, devido as vantagens que oferece;

- **Anaconda**: O Anaconda é uma distribuição livre e open-source das linguagens de programação *Python* e R para computação científica que visa simplificar a gestão e implantação de pacotes. As versões dos pacotes são geridas pelo sistema de gestão de pacotes conda. A distribuição Anaconda vem com mais de 1.500 pacotes A grande diferença entre a conda e o gestor de pacotes pip está na forma como as dependências de pacotes são geridas, o que é um desafio significativo para a ciência de dados Python e a razão pela qual a conda existe.

- **NLTK**: o *Natural Language Toolkit*, é a melhor biblioteca da categoria para tarefas de análise de texto. Segundo (http://www.nltk.org/) o NLTK é uma plataforma líder na construção de programas *Python* para trabalhar com dados da linguagem humana. NLTK fornece interfaces como o *WordNet*, juntamente com um conjunto de bibliotecas de processamento de texto para classificação, *tokenization*, *stemming*, *wrappers* para bibliotecas de NLP, entre outros. Escrito pelos criadores do NLTK, este *Natural Langua Tooklit* guia o leitor através dos fundamentos da escrita de programas Python, categorizando texto, analisando estrutura linguística, entre outros.

- **R e RStudio**: R é uma linguagem e um ambiente para computação estatística e gráficos. R fornece diversas de técnicas estatísticas e gráficas. Um dos pontos fortes de R é a facilidade com que se podem produzir gráficos de grande qualidade e bem desenhados,

incluindo símbolos matemáticos e fórmulas onde for necessário. *RStudio* é um software livre de ambiente de desenvolvimento integrado para R.

2.9 RELATED WORK

Hoje em dia, é vital para as organizações saber quais as opiniões dos consumidores em relação aos seus produtos ou serviços. No passado, estas organizações utilizavam métodos mais tradicionais como estudos, votações e discussões de grupo sempre que necessitavam de saber a opinião pública ou do consumidor. As aplicações de Análise de Sentimento [CN20] e *Text Mining* [ON19] têm proliferado em diversos domínios para os mais diversos campos, como *sites* de notícias [CN20], turismo[Ala+19], redes sociais, saúde, política [Liu12b], produtos e serviços para o consumidor [VFB15] entre outros. Sob a forma de tanto para dados estruturados[Bal17] como não estruturados [WZW19]. Atualmente as redes sociais são uma fonte de informação e que pode ser muito relevante para os mais diversos casos. Segundo o trabalho de [WZW19] através da análise dos *tweets* é possivel perceber a polaridade sentimental do texto, mas também preservar a semântica do texto não estruturado. A rede social *Twitter* é muito utilizada para recolha de informação, como no caso do trabalho [AB16], onde através análise dos dados do *Twitter* relacionados com duas empresas transportadoras, e aplicando análise de sentimento foi possível calcular o *Net Sentiment Score* que permite a essas empresas medir a satisfação dos seus clientes.
Uma outra aplicação destas técnias foi o caso da *Amazon* [Cas+17], onde é possível observar que através de métodos de *Machine Learning* obtiveram melhor desempenho quando lidando com dados não lineares e não estruturados e conseguiram prever o sucesso de produtos disponível na *Amazon* através do *feedback* dos utilizadores. Esta extração da opinião pode servir para identificadar a polaridade de sentimento (positiva ou negativa) [CN20] [ON19]. *Amazon* tem investido cada vez mais melhorar os seus produtos e serviços num estudo [Bal17] em que realizam este tipo de tarefas com dados estruturados, a mostra que as técnicas de processamento de linguagem natural e *Text Mining* podem ser utilizadas para identificar as principais características de um produto.
Olhando para o contexto do problema em estudo é possível encontrar casos ainda mais similares, onde são aplicadas as técnicas de Análise de Sentimento, *Text Mining*, processamento de linguagem natural como os casos de [ON19][VFB15] e o caso da empresa Telstra. Um caso de aplicação de Análise de Sentimento que pode ser considerado mais relevante no contexto deste projeto é o caso da empresa Telstra. Esta empresa de telecomunicações afirma que o envolvimento e análise do cliente na indústria dos *contact centres* é fundamental para uma organização de sucesso. De modo a avaliar o envolvimento do cliente definiram uma métrica principal chamada de NPS (*network promoter score*"). Cada cliente foi separado em três categorias onde conforme o seu nível de envolvimento seriam optadas diferentes

estratégias de *marketing*. Todos estes diferentes tipos de recolha de informação e análise do sentimento permite que sejam mais facilmente definidas estratégias e/ou perceber melhor o *feedback* dos consumidores.

2.10 DESAFIOS EXISTENTES

Após efetuar um levantamento do estado da arte consegue-se perceber mais concretamente quais os desafios existentes para este projeto de dissertação. Um dos maiores desafios para a obtenção de sucesso no decorrer deste projeto são os dados. Todo o projeto está bastante comprometido por este aspeto. Em primeiro lugar é expectável o acesso aos dados retirados do serviço ao cliente na área de telecomunicações, dados estes que serão numa quantidade bastante volumosa. Mas este aspeto não se resume apenas à grande quantidade de dados, outro fator é o facto de os dados serem relevantes para o projeto, visto que todos ou quase todos os dados que irão ser fornecidos vão ser dados não estruturados torna a análise dos mesmos um processo mais demoroso e necessitam de uma análise mais cuidadosa e um tratamento mais intensivo. Esta dependência dos dados realmente compromete e define o projeto, visto que todo este tema se insere na área de NLP apresenta essencialmente várias barreiras linguísticas. Estes desafios podem surgir em vários meios, como a utilização de abreviaturas, hifenização e uma má escrita por parte de quem escreveu os dados, entre outros. Apesar de parte destas técnicas poderem ser aplicadas a outros contextos e a outras línguas, existe a necessidade concreta em aplicar ao cenário específico da NOS e da língua portuguesa. É de ter em conta que todos estes dados sofrem de intervenção humana maioritariamente, sendo que estão bastantes sujeitos a erros e redundância. Finalmente, sendo NLP uma área em constante desenvolvimento várias técnicas estão sendo melhoradas diariamente e surgindo novos métodos, o que introduz mais um grau de dificuldade ao projeto.

3

METODOLOGIA

Com o intuito de idenfiticar os artigos relevantes, de modo a compreender os conceitos fundamentais para a dissertação, é necessário fazer uma pesquisa prévia e uma seleção dos artigos mais relevantes. Com esse intuito foram definidas que plataformas seriam utilizadas para encontrar os artigos que fosse mais relevants para a dissertação, plataformas como *Google Scholar, ScienceDirect, Semantic Scholar*, *Google , Web of Science* e ainda o repositório Uminho. Através da leitura dos títulos e *abstract* dos artigos, nem todos se tornaram relevantes para este projeto, para além desses dois aspetos foi também tido em conta o número de citações de cada artigo.

3.1 DESIGN SCIENCE RESEARCH

A pesquisa em sistemas de informação diz respeito a pessoas, organizações e tecnologia [Ven14]. O DSR (*Design Science Research*) é um procedimento de pesquisa para produzir uma construção inovadora destinada a resolver problemas enfrentados no mundo real e, com isso, contribuir para a teoria da disciplina em que é aplicado [KLS93].
Saunders et al. (2009) explicaram que, do ponto de vista de DSR, o principal objetivo da pesquisa de gestão acadêmica é desenvolver conhecimentos válidos para apoiar a resolução de problemas organizacionais. No paradigma de *Design Science*, o conhecimento e o entendimento de um determinado problema e a sua solução são alcançados através da construção da aplicação definida pelo artefacto desenhado [Ese+04]. As orientações aqui apresentadas são utilizadas como um enquadramento para o projecto manter o foco da investigação e resultados válidos. Em 2004 Hevner et al. definiu sete *guidelines* para o *Design Science in Information Systems Research* sendo elas:

1. **Design como um artefacto**- Os autores das diretrizes definem o artefato de TI como uma forma de construção, modelo, um método ou um instanciação. Esta é uma definição ampla de artefato de TI, mas nesta definição não estão incluídas pessoas ou elementos das organizações, para manter o foco no elemento tecnologias inovadoras.

Neste sistema o artefato são modelos de análise em documentos de texto e utilização de técnicas de *Text Mining*;

2. **Relevância do problema**- DSR conta com o artefato para construir um método de solução de um problema. Um problema é definido como "A diferença entre um estado de objetivo e o estado atual do sistema" [Ese+04]. O objetivo é desenvolver soluções tecnológicas para problemas, nesta fase foi efetuado o levantamento e interpretação do problema.

3. **Avaliação do *design***- A utilidade, qualidade e eficiência de um artefato de design deve ser rigorosamente demonstrada através de métodos de avaliação bem executados. O artefato pode ser avaliado em relação a: funcionalidade, completude, consistência, precisão, desempenho, confiabilidade, usabilidade, adequação à organização ou outros atributos relevantes [Ese+04] . Esta passo focou-se em verificar a viabilidade das técnicas utilizadas e a confirmação dos resultados obtidos.

4. **Contribuições da pesquisa**- Um DSR eficaz deve fornecer contribuições claras e verificáveis nas áreas de artefatos de *design*, fundações de design e/ou metodologias de *design*. Neste projecto as contribuições para a base de conhecimento são o próprio artefacto, e os resultados das avaliações.

5. **Rigor da pesquisa**- DSR baseia-se na aplicação de métodos rigorosos tanto na construção como na avaliação do artefacto. Neste ponto foi descrevido o processo de pesquisa utilizado nesta dissertação.

6. ***Design* como um Processo de Pesquisa**- A busca de um artefato eficaz requer a utilização dos meios disponíveis para atingir os fins desejados, satisfazendo ao mesmo tempo as leis do ambiente do problema. Nesta etapa foi efetuado um levantamento do estado da arte existente nas áreas desta dissertação.

7. **Comunicação da Pesquisa**- DSR deve ser apresentada de forma eficaz, tanto para o público orientado para a tecnologia como para o público orientado para a gestão. A principal comunicação desta pesquisa será esta tese. A tese será dirigida a um público mais acadêmico.

3.2 CRISP-DM

Nesta fase são descritas as metodologias utilizadas nesta dissertação. Em primeiro lugar, é abordada a metodologia de investigação, em seguida, é descrita a metodologia prática. O fator que condicionou a escolha desta metodologia em detrimento de outras, como por exemplo SEMMA e PMML , é o facto de esta metodologia ser mais completa, sendo que

ofereces inúmeras vantagens tais como: menores custos de execução; maior exequibilidade dos projetos; maior viabilidade dos projetos e maior rapidez [AS14]. Um outro fator inerente à escolha da metodologia foi a necessidade de seguir uma metodologia que possibilitasse abordar as várias fases inerentes a um projeto de *Text Mining*. Esta metodologia contempla seis fases que são flexíveis [Lau12] que formam o ciclo de vida do processo descrito na metodologia CRISP-DM estão representadas na Figura 3.1 bem como as relações entre as mesmas.

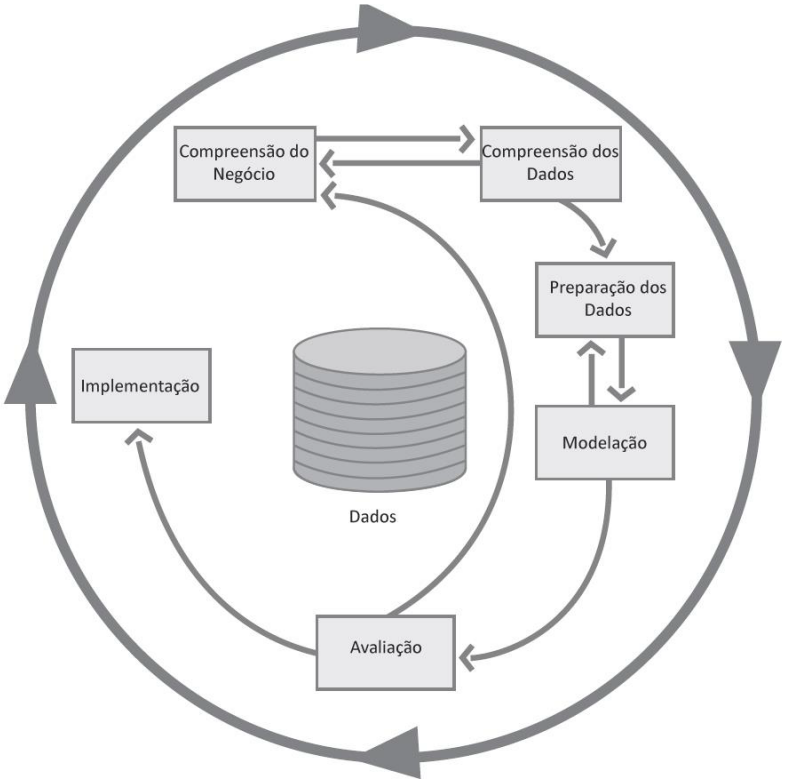


Figure 3.1: Ciclo de vida e fases da metodologia CRISP-DM, Chapman et al., 2000

A metodologia CRISP-DM é constituída por seis fases principais: compreensão do negócio, compreensão dos dados, preparação dos dados, modelação, avaliação e implementação. Na fase de compreensão do negócio, é expectável que se perceba o propósito pelo qual o projeto em causa está a ser desenvolvido, e qual a razão que motiva a utilização de *data mining* ou *Text Mining* para esse propósito. Esta compreensão do negócio, para esta dissertação em concreto, permite analisar as formas como serão aplicadas as técnicas de *Text Mining*.
A fase de compreensão dos dados tem como finalidade a análise dos mesmos, e caso ainda não tenha sido efetuada a recolha dos dados é possivel identificar como serão recolhidos. Uma vez que esta dissertação se encontra no âmbito de *Text Mining*, os dados recolhidos serão dados não estruturados ou em forma de texto.
Na fase de preparação dos dados é onde é expectável que sejam realizadas as diversas ativi-

dades de modo que os dados possam ser utilizados, mais concretamente nesta dissertação, dados textuais em atividades de *Text Mining* (por exemplo treinar e validar modelos). Esta fase é bastante importante pois envolve as etapas necessárias para que os dados textuais sejam estruturados, mas por vezes é necessário voltar à fase de preparação de dados para efetuar modificações devido a terem sido identificados novas necessidades a esse nível.
É durante a fase de modelação que se selecionam as várias técnicas de modelação e em que há um ajuste dos parâmetros, de forma a otimizar os resultados. Nesta fase ainda, podem ser efetuadas decisões relativas aos modelos, como a parametrização, e onde os modelos são treinados Nesta seleção é importante atender não só à adequação da técnica ao problema de *Text Mining*, mas também aos requisitos específicos que algumas destas técnicas têm.
Na fase de avaliação, avalia-se a utilidade dos modelos, reveem-se os passos executados e verifica-se se permitem atingir os objetivos do negócio. Os três passos desta fase são: Avaliação dos Resultados, a Revisão do Processo e a Determinação dos Próximos Passos. Não é apenas avaliado o desempenho dos modelos mas também efetuada a comparação de vários modelos com o intuito de determinar qual o mais adequado às necessidades identificadas na fase de compreensão do negócio.
Na fase de implementação, planeia-se a avaliação dos resultados, incluindo os passos e a forma de os executar; planeia-se a monitorização e manutenção; produz-se o relatório final e efetua-se a revisão do projeto. Para além destes passos, todo trabalho que foi desenvolvido nas etapas anteriores da metodologia deve ser implementado num ambiente real, que no caso em concreto do trabalho de dissertação, corresponde a um sistema que permita extrair conhecimento sobre dados não estruturados.

3.3 TAREFAS

Uma vez que para este projeto é utilizada a metodologia CRISP-DM, foram definidas as cinco tarefas principais. De notar ainda que a Tarefa 1 e Tarefa 2 foram desenvolvidas no âmbito da pré-dissertação e as restantes tarefas no âmbito da dissertação.

- **Tarefa 1**: Elaboração do plano de trabalhos- plano este que contém um resumo do que irá ser tratado, um enquadramento, os objetivos deste projeto, sendo o principal extrair informação relevante de dados não estruturados fornecidos pela NOS, técnicas que serão utilizadas, resultados esperados, a abordagem metodológica e por fim a calendarização.

- **Tarefa 2**: Escrita do relatório de pré-dissertação- nesta tarefa é necessário proceder a uma pesquisa científica sobre os conceitos relativos ao tema da dissertação, serão também pesquisados e investigados artigos da mesma área assim como identificar possíveis projetos empresariais que visem finalidades semelhantes e proceder a escrito do projeto do relatório de pré-dissertação.

- **Tarefa 3**: Desenvolvimento da componente prática da dissertação- Para tal, é necessário seguir a metodologia CRISP-DM, pelo que são definidas as seguintes tarefas:
  - Tarefa 3.1: Compreensão do negócio;
  - Tarefa 3.2: Compreensão dos dados;
  - Tarefa 3.3: Preparação dos dados;
  - Tarefa 3.4: Modelação;
  - Tarefa 3.5: Avaliação;
  - Tarefa 3.6: Implementação;
- Tarefa 4 Elaboração de documentação científica.
- Tarefa 5: Escrita da dissertação: é expectável que esta tarefa seja desenvolvida ao longo da componente prática da dissertação.

4

CONCLUSÕES E FUTURE WORK

A inteligência artificial já tem sido usada para melhorar a experiência do cliente na área das telecomunicações. Este trabalho focou os principais problemas da atualidade, onde um grande conjunto de dados é recolhido diariamente pelas empresas de telecomunicações, deste modo, foi necessário pensar na dimensão deste problema e num modo para o resolver. Todas as organizações tentam ganhar uma vantagem tecnológica sobre os concorrentes através do desenvolvimento das suas tecnologias. Este estudo vai ser aplicado aos dados de serviço de apoio ao cliente para extrair informação relevante que permita conhecer melhor o cliente e fornecer melhores serviços.

Após ter sido terminado este relatório de pré-dissertação irá ser feita uma análise exploratória da qualidade dos dados e uma limpeza dos mesmos, o que por si só é uma tarefa bastante díficil devido ao elevado volume de dados e ao facto que se tratam dados não-estruturados. Esta etapa é bastante relevante, talvez a mais importante e demorosa de todas, visto que todas as etapas seguintes estão dependentes desta. Para começar serão tratados e filtrados os dados de forma a remover muitos dados que não sejam úteis para futura extração e análise. Para proceder a essa extração irão ser aplicadas técnicas de processamento de linguagem natural como *keyword extration*, remoção de *stop-words*, entre outros. Estes ”novos” dados extraídos irão utilizados para a construção de modelos analíticos. Numa segunda fase, esses “novos” dados irão ser analisados através de modelos de *Machine Learning* e irá ser feita também uma análise de sentimento desses dados de modo a que se consiga tirar conclusões pertinentes e que ajudem no âmbito deste projeto da NOS.

Finalmente, estes modelos serão avaliados através de diferentes métricas, com o intuito de perceber a veracidade dos resultados obtidos por cada modelo, quais os melhores modelos e obter uma melhor interpretação dos resultados.

BIBLIOGRAPHY

[Adm13] Seminários Administração. "Inovação e comportamento setorial : uma análise das empresas participantes do Prêmio FINEP de Inovação 2010". In: (2013).

[Ala+19] Dini Turipanam Alamanda et al. "Sentiment Analysis Using Text Mining of Indonesia Tourism Reviews via Social Media". In: *GI Social Sciences Forum*. 2019.

[All95] Allen. "Artificial Intelligence: A New Synthesis". In: (1995).

[AB16] Sonia Anastasia and Indra Budi. "Twitter sentiment analysis of online transportation service providers". In: *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE. 2016, pp. 359–365.

[AS14] Ana Azevedo and Manuel Filipe Santos. "KDD , SEMMA AND CRISP-DM : A PARALLEL OVERVIEW". In: June (2014).

[Bal17] Sinduja Balasubramanian. "Text mining on Amazon reviews to extract feature based feedback". In: (2017).

[Bar12] Thomas Bardoel. "Comparing n-gram frequency distributions Explorative research on the discriminative power of n-gram frequencies in newswire corpora". In: *Comparing n-gram frequency distributions Explorative research on the discriminative power of n-gram frequencies in newswire corpora* (2012). URL: http://arno.uvt.nl/show.cgi?fid=127240.

[BL06] David M Blei and John D Lafferty. "Dynamic topic models". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 113–120.

[Bro10] Gavin Brown. "Ensemble Learning". In: (2010). Ed. by Claude Sammut and Geoffrey I. Webb, pp. 312–320. DOI: 10.1007/978-0-387-30164-8_252. URL: https://doi.org/10.1007/978-0-387-30164-8%5C_252.

[BKM02] Heide Brücher, Gerhard Knolmayer, and Marc-andré Mittermayer. "Document Classification Methods for Organizing Explicit Knowledge Document Classification Methods for Organizing Explicit Knowledge Summary". In: 41.140 (2002), pp. 0–25.

[Buc88] Salton & Buckley. "Learning to Classify Text Using Support Vector Machines". In: *Learning to Classify Text Using Support Vector Machines* (1988).

[Bus05] Stephan Busemann. "Automated Text Summarization". In: (2005).

[Cas+17] Mauro Castelli et al. "An expert system for extracting knowledge from customers' reviews: The case of Amazon. com, Inc." In: *Expert Systems with Applications* 84 (2017), pp. 117–126.

[CN20] Alaleh Sadat Hosseini Charyani and Alireza Norouzi. "A Recommender System Based On Collaborative Filtering Using Polarity Improvement in Sentiment Analysis". In: *Majlesi Journal of Telecommunication Devices* 9.1 (2020), pp. 9–15.

[Erc06] Gönenç Ercan. "Automated text summarization and keyphrase extraction". In: *Unpublished MSc thesis, Bilkent University* (2006).

[Ese+04] S Ystems R Esearch et al. "DESIGN SCIENCE IN INFORMATION". In: 28.1 (2004), pp. 75–105.

[Gaz85] Pullum e Sag Gazdar, Klein. "Design Science in Information Systems Research". In: (1985).

[GS96] Ralph Grishman and Beth M Sundheim. "Message understanding conference-6: A brief history". In: (1996).

[Gro86] Jones & Webber Grosz. "Artificial Intelligence: A New Synthesis". In: (1986).

[GL+09] Vishal Gupta, Gurpreet S Lehal, et al. "A survey of text mining techniques and applications". In: *Journal of emerging technologies in web intelligence* 1.1 (2009), pp. 60–76.

[GL09] Vishal Gupta and Gurpreet S. Lehal. "A survey of text mining techniques and applications". In: *Journal of Emerging Technologies in Web Intelligence* 1.1 (2009), pp. 60–76. ISSN: 17998859. DOI: 10.4304/jetwi.1.1.60-76.

[Hal+10] G Brent Hall et al. "Community-based production of geographic information using open source software and Web". In: 8816 (2010). DOI: 10.1080/13658810903213288.

[HKK01] Eui-hong Sam Han, George Karypis, and Vipin Kumar. "Text Categorization Using Weight Adjusted k -Nearest Neighbor Classification ?" In: (2001).

[Ho98] Tin Kam Ho. "Nearest Neighbors in Random Subspaces". In: Lecture Notes in Computer Science 1451 (1998). Ed. by Adnan Amin et al., pp. 640–648. DOI: 10.1007/BFb0033288. URL: https://doi.org/10.1007/BFb0033288.

[Hud84] Hudson. "Word Grammar: Perspectives on a Theory of Language Structure". In: (1984).

[Joa98] Thorsten Joachims. "Text Categorization with Support Vector Machines : Learning with Many Relevant Features". In: (1998), pp. 2–7.

[JFM96] Thorsten Joachims, Dayne Freitag, and Tom Mitchell. "WebWatcher : A Tour Guide for the World Wide Web". In: September (1996).

[JM] Daniel Jurafsky and James H Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition". In: ().

[KLS93] Eero Kasanen, Kari Lukka, and Arto Siitonen. "The constructive approach in management accounting research". In: *Journal of management accounting research* 5.1 (1993), pp. 243–264.

[Kim+14] Jong Woo Kim et al. "Application of Decision-Tree Induction Techniques to Personalized Advertisements on Internet Storefronts Application of Decision-Tree Induction Techniques to Personalized Advertisements on Internet Storefronts". In: 4415 (2014). DOI: 10.1080/10864415.2001.11044215.

[Lam+16] Guillaume Lample et al. "Neural Architectures for Named Entity Recognition". In: *Neural Architectures for Named Entity Recognition* (2016). URL: https://arxiv.org/pdf/1603.01360.pdf%EF%BC%9ANeural.

[Lau12] Nuno Lau. "A METHODOLOGY FOR CREATING INTELLIGENT WHEELCHAIR USERS ' PROFILES". In: (2012), pp. 171–179. DOI: 10.5220/0003749701710179.

[LKR09] Lam Hong Lee, Vish P Kallimani, and Rajprasad Rajkumar. "Text Document Pre-Processing Using the Bayes Formula for Classification Based on the Vector Space Model Text Document Pre-Processing Using the Bayes Formula for Classification Based on the Vector Space Model". In: June 2014 (2009). DOI: 10.5539/cis.v1n4p79.

[LV09] Wenhui Liao and Sriharsha Veeramachaneni. "A Simple Semi-supervised Algorithm For Named Entity Recognition". In: June (2009), pp. 58–65.

[Liu12a] Bing Liu. "Sentiment Analysis and Opinion Mining". In: Synthesis Lectures on Human Language Technologies (2012). DOI: 10.2200/S00416ED1V01Y201204HLT016. URL: https://doi.org/10.2200/S00416ED1V01Y201204HLT016.

[Liu12b] Bing Liu. "Sentiment analysis and opinion mining". In: *Synthesis lectures on human language technologies* 5.1 (2012), pp. 1–167.

[LC11] Wei Liu and Sanjay Chawla. "Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets". In: (2011).

[Luh58] Hans Peter Luhn. "A Business Intelligence System". In: *A Business Intelligence System* (1958).

[Mar18] B Marr. "The Internet of Things (IOT) will be massive in 2018: here are the 4 predictions from IBM". In: *Forbes. Recuperado de https://www. forbes. com/sites/bernardmarr/2018/01/04/the-internet-ofthings-iot-will-be-massive-in-2018-here-are-the-4-predictions-from-ibm* (2018).

[Mic68] Donald Michie. ""Memo" functions and machine learning". In: *Nature* 218.5136 (1968), pp. 19–22.

[MT04] Rada Mihalcea and Paul Tarau. "TextRank : Bringing Order into Texts 132547658 $ 9 @ 2BADC EF8 " GHEPI RTSVU ¨ Q WFX ' Y acb e fhgpi d XqYsrtb e 1u254 R 8 v v". In: July (2004).

[Mit97] Tom M Mitchell. "Does machine learning really work?" In: *AI magazine* 18.3 (1997).

[MBR10] Ashwin Mohan, Ibrahim M Baggili, and Marcus K Rogers. "Authorship attribution of SMS messages using an N-grams approach". In: *Tech. Rep., 7* (2010).

[NS12] Andrew J. Nathan and Andrew Scobell. "Computational Linguistics and Intelligent Text Processing: 14th". In: *Foreign Affairs* 91.5 (2012), pp. 1–58. ISSN: 00157120. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[Oga16] Kennedy Odhiambo & Ogada. "N-grams for Text Classification Using Supervised Machine Learning Algorithms". In: *N-grams for Text Classification Using Supervised Machine Learning Algorithms* (2016).

[ON19] Kingsley A Ogudo and Dahj Muwawa Jean Nestor. "Sentiment Analysis Application and Natural Language Processing for Mobile Network Operators' Support on Social Media". In: *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*. IEEE. 2019, pp. 1–10.

[OCA13] N. Oliveira, P. Cortez, and N & Areal. "On the predictability of stock market behavior using StockTwits sentiment and posting volume. In Progress in artificial intelligence: 16th Portuguese Conference on Artificial Intelligence, EPIA 2013: proceedings (pp. 355–365). Springer." In: *On the predictability of stock market behavior using StockTwits sentiment and posting volume. In Progress in artificial intelligence: 16th Portuguese Conference on Artificial Intelligence, EPIA 2013: proceedings (pp. 355–365). Springer.* (2013). URL: http://hdl.handle.net/1822/31408.

[OCA17] N. Oliveira, P. Cortez, and N. & Areal. "The impact of microblogging data for stock market prediction: Using Twitter to predict returns, volatility, trading volume and survey sentiment indices. Expert Systems with Applications". In: *The impact of microblogging data for stock market prediction: Using Twitter to predict returns, volatility, trading volume and survey sentiment indices. Expert Systems with Applications*, (2017). URL: https://doi.org/https://doi.org/10.1016/j.eswa.2016.12.036.

[PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment Classification using Machine Learning Techniques". In: *CoRR* cs.CL/0205070 (2002). URL: https://arxiv.org/abs/cs/0205070.

[Pas17] Hristo Spassimirov Paskov. "LEARNING WITH N-GRAMS: FROM MASSIVE SCALES TO COMPRESSED REPRESENTATIONS". In: *LEARNING WITH N-GRAMS: FROM MASSIVE SCALES TO COMPRESSED REPRESENTATIONS* (2017). URL: https://web.stanford.edu/%7B~%7Dhastie/THESES/hristo%7B%5C_%7Dthesis.pdf.

[Por01] M.F. Porter. "Snowball: A language for stemming algorithms". In: *Snowball: A language for stemming algorithms* (2001).

[Rab89] L. R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989).

[Raj11] J. D. Rajaraman, A. & Ullman. "Mining of Massive Datasets". In: *Mining of Massive Datasets* (2011). URL: https://dl.acm.org/doi/book/10.5555/2124405.

[RR15] Kumar Ravi and Vadlamani Ravi. "A survey on opinion mining and sentiment analysis: tasks, approaches and applications". In: *Knowledge-Based Systems* 89 (2015), pp. 14–46.

[Sch92] Edgar H. Schein. "TOKENIZATION AS THE INITIAL PHASE IN NLP". In: *Japanese Society of Biofeedback Research* 19.1979 (1992), p. 1992. DOI: 10.20595/jjbf.19.0_3.

[Ste+04] Mark Steyvers et al. "Probabilistic author-topic models for information discovery". In: (2004). Ed. by Won Kim et al., pp. 306–315. DOI: 10.1145/1014052.1014087. URL: https://doi.org/10.1145/1014052.1014087.

[Tan+05] Lorraine Tanabe et al. "GENETAG : a tagged corpus for gene / protein named entity recognition". In: 7 (2005), pp. 1–7. DOI: 10.1186/1471-2105-6-S1-S3.

[TR13] Paulo Trigueiros and Fernando Ribeiro. "A Comparative Study of Different Image Features for Hand Gesture Machine Learning". In: (2013), pp. 51–61.

[Ven14] John R Venable. "Design Science Research Post Hevner et al .: Criteria , Standards , Guidelines , and Expectations Design Science Research Post Hevner et al : Criteria , Standards , Guidelines , and Expectations". In: June 2010 (2014), pp. 0–16. DOI: 10.1007/978-3-642-13335-0.

[VFB15] Nur Azizah Vidya, Mohamad Ivan Fanany, and Indra Budi. "Twitter sentiment to analyze net brand reputation of mobile phone providers". In: *Procedia Computer Science* 72 (2015), pp. 519–526.

[WZW19] Wenjun Wang, Feng Zhou, and Xiao Wei. "The STViewer, a Visual Method with Sentiment Analysis: Retrieve Information and Visualize Social Media Text Better". In: *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*. 2019, pp. 50–56.

[WC06] Xing Wei and W Bruce Croft. "LDA-based document models for ad-hoc retrieval". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 2006, pp. 178–185.

[WFH] Ian H Witten, Eibe Frank, and Mark A Hall. *Data Mining Practical Machine Learning Tools and Techniques Fourth Edition*. ISBN: 9780128042915.

[Yu+18] L.-C. Yu et al. "Refining Word Embeddings Using Intensity Scores for Sentiment Analysis". In: *Refining Word Embeddings Using Intensity Scores for Sentiment Analysis* (2018). URL: https://ieeexplore.ieee.org/abstract/document/8241844.

[Zha88] Zhang. "Definitions and Sciences of information. Information Processing & Management". In: *Definitions and Sciences of information. Information Processing & Management,* (1988).

[ZE13] Shaodian Zhang and Noémie Elhadad. "Unsupervised biomedical named entity recognition : Experiments with clinical and biological texts". In: *Journal of Biomedical Informatics* 46.6 (2013), pp. 1088–1098. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2013.08.004. URL: http://dx.doi.org/10.1016/j.jbi.2013.08.004.

[Zor95] Zorrinho. "Gestão da Informação. Condição para Vencer". In: *Gestão da Informação. Condição para Vencer* (1995).