UNIVERSIDADE FEDERAL DE PERNAMBUCO

Centro de Informática

Graduação em Engenharia da Computação

Trabalho da Graduação

Título: Um classificador para Análise de Sentimento a partir de Comentários sobre Produtos Eletrônicos

Silvio Romero de Santana Junior

Recife, dezembro de 2018

UNIVERSIDADE FEDERAL DE PERNAMBUCO

Centro de Informática

Graduação em Engenharia da Computação

Trabalho da Graduação

Título: Um classificador para Análise de Sentimento a partir de Comentários sobre Produtos Eletrônicos

Silvio Romero de Santana Junior

Trabalho de Graduação apresentado ao centro de Informática da Universidade Federal de Pernambuco como requisito para obtenção do Grau de Bacharel em Engenharia da Computação

Orientador: Ricardo Bastos C. Prudêncio

Recife, dezembro de 2018

Dedicatória

Dedico esse trabalho a minha mãe Magda Patrícia Coelho Ramalho, ao meu pai Marcos Wagner de Melo e a minha avó Maria Veronica Bezerra.

Agradecimentos

Gostaria de agradecer a minha família, que sem dúvida alguma é o meu maior apoio, sem eles eu nada seria. Em especial, minha mãe Magda Patrícia Coelho Ramalho e meu pai Marcos Wagner de Melo.

Aos professores da Universidade Federal de Pernambuco, em especial, aos professores Ricardo Bastos C. Prudêncio, Odilon Maroja da Costa Pereira Filho e José Américo de Miranda Neto, que fizeram uma grande diferença em minha graduação.

Agradeço também à Universidade Federal de Pernambuco, que me ofereceu ensino e estrutura de qualidade em minha jornada nessa etapa da minha vida.

Resumo

A quantidade de pessoas que compram em lojas online está cada vez maior. Entre os produtos comprados, os mais populares são os eletrônicos, em especial, os telefones celulares. Produtos como esses podem ser comprados em diversos sites na internet, podendo ser esses sites genéricos de venda (vendem vários tipos de produto), ou sites especializados em eletrônicos, como os sites das próprias empresas que fabricam marcas populares de celulares.

Para esses sites, é importante saber quais produtos estão sendo bem ou mal avaliados, ou seja, saber o quão positiva ou negativa é a opinião dos usuários a respeito dos produtos comercializados. Muitos comentários negativos, por exemplo, podem indicar que um produto em especial pode ter apresentando algum problema inesperado ou que alguma característica especifica não foi bem aceita pelos compradores do produto. Nesse último caso, essa característica pode ser removida nos próximos modelos do produto. Da mesma forma, comentários positivos podem indicar características bem aceitas pelos compradores.

Comentários sobre celulares são populares na internet, já que praticamente todo site de vendas disponibiliza aos seus usuários acesso a uma base de comentários de outras pessoas, assim como a opção de comentar sobre um ou mais produtos. Infelizmente, ler todos os comentários é uma tarefa inviável, a depender da quantidade de mensagens deixadas pelos usuários.

Este trabalho apresenta uma abordagem para processar essas informações e extrair sentimentos polarizados (positivos ou negativos) usando ferramentas de pré-processamento e classificação de texto. Os experimentos realizados comparam três (3) diferentes classificadores e diferentes técnicas de tratamento de texto. O objetivo foi identificar quais técnicas de classificação estão mais aptas a serem usadas no processo de classificação de comentários, assim como, testar o desempenho dos classificadores Naive Bayes, Suport Vector Machine e Random Forest.

Palavras Chave: Analise de sentimento, classificadores, processamento de linguagem natural, pré-processamento de texto.

Sumário

1. Introdução 12

1.1. Contexto e Problema 12

1.2. Motivação e Solução 13

1.3. Organização do Projeto 14

2. Análise de Sentimento – Conceitos básicos 16

2.1. Entidades e suas característica 16

2.2. Opiniões e Fatos 16

2.3. Polaridade 17

2.4. Níveis de análise de sentimento 17

2.5. Aplicações 18

3. Técnicas para Construção de Classificadores de Sentimento 21

3.1. Construção de classificadores de texto 21

3.1.1. Aprendizagem de máquina supervisionada e não supervisionada . 22

3.1.2. Abordagem baseada em Léxicos Polarizados 22

3.2. Técnicas de pré-processamento 23

3.2.1. Tokenização 24

3.2.2. Eliminação de Stopwords 24

3.2.3. Processo de Stemming 25

3.2.4. Uso de N-Grama 26

3.3. Construção de Classificadores baseados em Aprendizagem Supervisionada 28

3.3.1. Naive Bayes 29

3.3.2. Suport Vector Machine (SVM) 31

3.3.3. Random Forest 33

3.3.4. Métricas de avaliação 35

3.4. Ferramentas e Bibliotecas de Apoio 37
3.4.1. WEKA 38
3.4.2. Biblioteca NLTK 39
3.4.3. Linguagem PYTHON 40
3.5. Desafios e limitações 40
4. Trabalho Realizado 41
4.1 Visão geral 41
4.2. Corpus de Comentários 42
4.3. Fase de pré-processamento 42
4.3.1. Técnicas Obrigatórias 43
4.3.2. Técnicas Opcionais 48
4.4 Indução de Classificadores usando WEKA 51
5. Experimentos e resultados 52
5.1. Instâncias de arquivos de treinamento e testes 52
5.2. Realizando experimentos com WEKA 53
5.3. Resultados 54
5.3.1. Resultados do classificador Naive Bayes 54
5.3.2. Resultados do classificador Suport Vector Machine 55
5.3.3. Resultados do classificador Random Forest 56
5.3.4. Comparando classificadores 57
5.3.5. Considerações finais 58
6. Conclusão 60
6.1. Contribuições e Dificuldades 60
6.2. Trabalhos futuros 60
Referências Bibliográficas 62

Lista de Imagens

Imagem 1 - (a) Espaço linearmente separável, (b) Espaço não linearmente separável 31
Imagem 2 - Vetores de suporte e margem de separação 31
Imagem 3 - Hiperplano após transformação linear 32
Imagem 4 - Aplicação de função Kernel 32
Imagem 5 - Árvore de decisão para jogo de futebol 33
Imagem 6 - Random Forest 34
Imagem 7 - Gráfico da área sobre a curva 37
Imagem 8 - Exemplo de arquivo ".arff" 38
Imagem 9 - Escolhendo um classificador 39
Imagem 10 - Resultados da classificação WEKA 39
Imagem 11 - Arquitetura geral do projeto 41
Imagem 12 - Exemplos da base de documentos 42
Imagem 13 - Fluxograma para algoritmo de pré-processamento 43
Imagem 14 - Processo de tokenização 44
Imagem 15 - Retirada de caracteres e palavras irrelevantes 45
Imagem 16 - Representação bag-of-words 47
Imagem 17 - Criação da estrutura bag-of-words 47
Imagem 18 - Representação do arquivo de saída para nosso pré-processamento 48
Imagem 19 - a) Criação das instâncias de treinamento, b) Mudando a forma de representar as features, c) Definindo os atributos no arquivo arff, ... 48
Imagem 20 - a) Retirada dos advérbios da lista de stopwords, b) Retirada das stopwords dos documentos de texto, c) Definindo tipos de advérbios 50
Imagem 21 - Aplicação de stemming nas palavras 50
Imagem 22 - Representando textos com N-gramas 51
Imagem 23 - Representação do nome da Instância de arquivo de teste 52
Imagem 24 - Instâncias de Teste 53
Imagem 25 - Escolha: a) Naive Bayes, b) SVM, c) Random Forest 53

Lista de tabelas

Tabela 1 - Stemming de palavras..... 25
Tabela 2 - Retirada de caracteres e palavras irrelevantes..... 45
Tabela 3 - Resultados para o classificador Naive Bayes..... 54
Tabela 4 - Resultados para classificador Suport Vector Machine..... 55
Tabela 5 - Resultados para o classificador Random Forest..... 56
Tabela 6 - Comparando os melhores resultados..... 57
Tabela 7 - Comparando os piores resultados..... 58

Lista de equações

Equação 1 - Cálculo da probabilidade de uma feature pertencer a uma classe 30
Equação 2 - Cálculo da precisão para a classe positiva 35
Equação 3 - Cálculo da cobertura 36
Equação 4 - Cálculo da Acurácia 36
Equação 5 - Cálculo da F-Measure 37

Lista de exemplos

Exemplo 1 - Tipos de sentimentos 16
Exemplo 2 - Opiniões e fatos 17
Exemplo 3 - Tipos de opiniões 17
Exemplo 4 - Níveis de Sentimento 18
Exemplo 5 - Identificação de tokens..... 24
Exemplo 6 - Exemplos de stopwords 24
Exemplo 7 - Perca de contexto para a mesma palavra..... 26
Exemplo 8 - Preservação do contexto para bigrama..... 26
Exemplo 9 - Preservação do contexto para bigrama sem stopwords..... 27
Exemplo 10 - Representação de documento em trigramas 28
Exemplo 11 - Detecção de features em sentenças..... 30
Exemplo 12 - Representação de tuplas no arquivo saída 46

1. Introdução

Com o advento da internet os consumidores ganharam novas formas de opinar sobre os produtos oferecidos pelo mercado. Agora eles podem avaliar, destacando pontos positivos e negativos de cada produto, podendo apresentar suas opiniões em fóruns, sites e até redes sociais [1]. Na seção 1.1 falamos um pouco mais sobre esse cenário e na seção 1.2 sobre a motivação que nos levou a trabalhar nesse contexto e solução proposta.

1.1. Contexto e Problema

As empresas de venda são favorecidas pelo novo modo de participação dos consumidores, já que podem ter uma ideia geral da impressão que seus produtos deixaram nos clientes. Através desse feedback, são capazes de identificar mais rapidamente características positivas e negativas em seus produtos, de modo que seus próximos lançamentos possam se beneficiar das qualidades mais admiradas e eliminar as características que foram menos aceitas pelos consumidores nos produtos anteriores.

Para potenciais clientes, é importante ter acesso as opiniões de outros clientes no momento de fazer sua escolha. Assim, podem ficar a par dos tópicos mais importantes referentes aos produtos visados: defeitos, comparações, qualidades etc. Assim, o consumidor terá acesso a uma base de informações que o ajudará a tomar a melhor decisão na hora da compra.

Atualmente, a maioria das empresas comerciais oferecem essas informações para seus consumidores no formato de uma base de dados com vários comentários escritos em linguagem natural, muitas vezes sem nenhuma estrutura pré-estabelecida ou tratamento para remover partes do texto que não são consideradas uteis (palavras com grafia errada por exemplo). Tendo em vista que o cliente tem liberdade para criar seu comentário como quiser, é possível ver todo tipo de manifestação de sentimento, como: ironia, sarcasmo, raiva, felicidade, tristeza entre outras. Também é possível observar pontuação exagerada, assim como símbolos conhecidos como *emoticons*, esses últimos muito utilizados em redes sociais.

Muitas vezes, para sintetizar a opinião de cada consumidor, são usadas medidas quantitativas para avaliar um produto, permitindo que o cliente possa dar uma nota ao item avaliado. Números de estrelas (1-5) e notas (1-10) são as formas mais comuns de se avaliar um produto. Esse método ajuda o consumidor a ter uma ideia da opinião geral dos outros usuários sem precisar ler todas as mensagens. Infelizmente, não é todo site que oferece mecanismos como esse para os consumidores, e não são todos os consumidores que usam esses mecanismos, mesmo quando estão disponíveis.

Com o objetivo de processar tantos comentários mais rapidamente e da maneira mais precisa possível, vários algoritmos foram desenvolvidos para tratar essas informações, permitindo que centenas, até milhares de comentários possam ser avaliados rapidamente. Assim, uma impressão geral do produto pode ser exposta para o consumidor, para que o mesmo não tenha que avaliar comentário por comentário, o que muitas vezes se torna uma tarefa inviável a depender da quantidade de mensagens a serem lidas.

Também são usadas técnicas de pré-processamento de texto para tratar os comentários e retirar informações inúteis. Dessa forma, os algoritmos de avaliação podem ser mais eficientes na hora do processamento. As técnicas de pré-processamento que são usadas dependem do contexto onde os comentários foram expressos, assim como dependem também de sua relevância para o desempenho do algoritmo de avaliação.

1.2. Motivação e Solução

Uma das abordagens mais populares para classificação de um documento (no nosso contexto, comentários são os documentos a serem classificados) é a tentativa de analisar sua polaridade. Quando um comentário apresenta uma polaridade, pode ser avaliado como positivo, negativo ou neutro. Mais à frente vamos nos aprofundar no conceito de polaridade.

Antes que os algoritmos de classificação iniciem o processo de classificação dos textos, é importante realizar uma etapa de pré-processamento do texto cujo objetivo é limpar os textos de palavras, termos ou símbolos irrelevantes, que não vão influenciar de forma positiva o posterior processamento

dos documentos pelos classificadores. Entre as técnicas de pré-processamento mais comuns estão as retiradas de stop-words e o stemming.

Nesse projeto utilizamos uma ferramenta de software que implementa os algoritmos de classificação mais populares. O WEKA[1] recebe como entrada documentos etiquetados com classes de sentimento (pos, neg, neut) e oferece os resultados da classificação usando métricas como: falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos, precisão, cobertura, F-Measure e AUC. Além disso oferece a opção de treinamento cruzado e holdout, dinamizando a forma como os classificadores são treinados.

O objetivo do projeto é construir e comparar classificadores de sentimento. Para isto, utilizamos a ferramenta WEKA para treinar classificadores e comparar os resultados que dependem do algoritmo usado e do pré-processamento utilizado no tratamento do texto. Através dessa comparação, foi possível definir quais são as técnicas mais relevantes para que se alcance um resultado satisfatório na classificação dos comentários. O corpus de comentários utilizado na construção dos classificadores consiste em uma base de dados pré-selecionada e classificada manualmente. O contexto escolhido para implementar o projeto foi o de sites de vendas de eletrônicos, especificamente, telefones celulares.

1.3. Organização do Projeto

No capítulo 1 apresentamos o projeto, seu contexto, problemas, motivações e solução proposta.

No capítulo 2 apresentamos conceitos básicos sobre Análise de sentimento.

No capítulo 3 falamos sobre classificadores de sentimento, tipos de aprendizagem, técnicas e abordagens. Vemos também ferramentas e bibliotecas de apoio utilizadas no projeto.

No capítulo 4 vemos detalhes sobre o projeto implementado, base de dados utilizada, técnicas de pré-processamento escolhidas e metodologia de desenvolvimento dos classificadores.

[1] https://www.cs.waikato.ac.nz/ml/weka/

No capítulo 5 falamos sobre os experimentos e resultados levando em consideração as técnicas aplicadas e classificadores testados.

No capítulo 6 apresentamos a conclusão do trabalho, com considerações sobre o processo e dificuldades encontradas. Por fim, sugerimos alguns trabalhos futuros que podem estender o trabalho aqui apresentado.

2. Análise de Sentimento – Conceitos básicos

Análise de sentimento é a área que estuda sentimentos presentes em textos. O problema fundamental em mineração de sentimento é descobrir quando sentimentos estão sendo expressos em textos e quais são esses sentimentos [2]

> *“ (1) Quando eu comprei esse celular esperava mais de seu desempenho. (2) Esse celular deixa muito a desejar. (3) Ótimo celular, muito rápido e bonito. (4) Esse é o melhor celular que eu já tive. ”*

Exemplo 1 - Tipos de sentimentos

Os comentários do exemplo 1 foram tirados do site da amazon.com [36] e são a respeito de um modelo de smartphone. A suposição inicial é que cada comentário como esse possui sentimentos expressos e que podem ser extraídos por meios computacionais. Nas sentenças 1 e 2 podemos observar que os sentimentos que podem ser extraídos são negativos, pois o escritor mostra sua insatisfação com o produto. Por outro lado, nas sentenças 3 e 4 o autor do texto mostra sua satisfação com o produto, assim, essas sentenças possuem valor positivo. Também é possível que uma sentença não expresse sentimentos.

2.1. Entidades e suas característica

Uma opinião é expressa sobre uma entidade. Uma entidade pode ser uma pessoa, produto, serviço ou organização. Em geral, opiniões podem ser manifestadas em fóruns, sites comerciais, redes sociais ou em qualquer outro ambiente que ofereça suporte para que os usuários deixem suas mensagens.

Uma opinião também pode ser expressa sobre uma característica de uma entidade. Na sentença 2 do exemplo 2, por exemplo, o autor do comentário elogia não o produto inteiro, mas, sim, uma de suas características (sua bateria).

2.2. Opiniões e Fatos

Uma opinião pode ser definida como um texto subjetivo que expressa um sentimento especifico. Já um fato é um texto objetivo que expressa uma

realidade. Ambos podem ser expressos sobre produtos, serviços, pessoas ou organizações [3].

“ (1) Eu já comprei 3 telefones em toda minha vida”. (fato) “(2) A bateria do meu celular dura bastante. ” (opinião)

Exemplo 2 - Opiniões e fatos

No exemplo 2, a sentença 1 é um fato já que não é uma mensagem subjetiva, ou seja, não se trata de uma visão pessoal e sim de uma realidade na vida do autor do comentário. Já na sentença 2, temos uma opinião, pois a durabilidade da bateria do celular do autor é um conceito abstrato, já que para pessoas diferentes um mesmo intervalo de tempo pode ser interpretado de modo diferente. Nesse caso, trata-se da opinião do autor.

Opiniões podem ser diretas ou comparativas. Na sentença 2 do exemplo 3, temos uma opinião direta, onde o autor elogia o produto de forma objetiva e clara. Na sentença 1, por outro lado, o autor elogia um produto através da comparação com outro produto, essa opinião é tida como opinião comparativa. Quando tentamos extrair sentimentos através de opiniões comparativas, precisamos saber quais são as entidades envolvidas ou extrair as entidades do texto de modo automático usando técnicas especificas.

“ (1) O Zenfone 3 tem uma tela melhor que o Samsung. ” *“ (2) Eu gosto bastante desse telefone. ”*

Exemplo 3 - Tipos de opiniões

2.3. Polaridade

A classificação de polaridade busca classificar semanticamente um texto em classes, que podem ser: positivas, negativa ou neutras. Nesse trabalho, apenas as classes positivas e negativas serão usadas. Como exemplo, podemos utilizar mais uma vez o exemplo 1, no qual a polaridade das sentenças 1 e 2 são ditas negativas, enquanto as sentenças 3 e 4 possuem polaridade positiva.

2.4. Níveis de análise de sentimento

Existem 3 níveis de análise de sentimento: nível de documento completo, de sentença ou de aspecto [4] [5]. No nível de documento [6], o texto como um todo pode ser classificado como positivo, negativo ou neutro. No nível de

sentença [7], o texto é dividido em sentenças, que têm suas polaridades calculadas individualmente. Já no nível de aspecto [8], são extraídos dos textos várias entidades e aspectos, e depois é calculada a polaridade para cada um deles.

"(1) Esse celular é bem barato, ótima aparência, boa compra.
(2) infelizmente, a tela é ruim, assim como a bateria.
(3) Voltarei a comprar essa marca, é muito boa"

Exemplo 4 - Níveis de Sentimento

A sentença 1 do exemplo 4 é positiva, já que o autor do texto avalia positivamente o produto, a sentença 2, por outro lado é negativa, já que o autor critica o produto. A sentença 3 também pode ser avaliada positivamente baseada no mesmo conceito que avaliou a primeira sentença. Então, a nível de sentença, podemos dizer que temos 2 sentenças positivas e 1 sentença negativa.

O documento como um todo pode ser dito positivo, já que o autor faz mais elogios do que críticas. Entretanto, isso é subjetivo, pois cada sentença avaliada positivamente ou negativamente pode possuir pesos diferentes dependendo das técnicas usadas para classificar o comentário.

Em relação aos aspectos, no exemplo acima são identificados vários aspectos e seus respectivos sentimentos: aparência (ótima), tela (ruim), marca (muito boa), preço (barato). Dessa forma, não avaliamos o produto diretamente, mas sim suas características. Nesse projeto, iremos trabalhar no nível de documento.

2.5. Aplicações

Redes sociais e mídias sociais

Analise de sentimento é amplamente usada em redes sociais, graças ao alto número de comentários disponíveis. Além disso, todos os tipos de pessoas usam redes sociais. Dessa forma, a variedade de comentários pode ser maior.

A grande variedade e quantidade de comentários favorece a criação de classificadores mais genéricos, capazes de lidar com tipos diferentes de

contextos. Outro ponto é que as redes e mídias sociais nos permitem coletar dados em linguagens diferentes [12][13].

Sites de venda

É nesse contexto em que o nosso projeto se enquadra. Com acesso a comentários de sites de vendas, é possível entender melhor o que os clientes querem. Esse feedback é muito importante para as empresas interessadas nos *reviews* (comentários de avaliação), pois a partir daí elas podem melhorar seus produtos ou serviços. Como já comentado aqui, os clientes também se interessam pelas opiniões dos outros usuários, dessa forma, os dois lados ganham [14][15].

Mercado financeiro

Através da análise de sentimentos em sites, redes sociais, fóruns e blogs podemos saber quais empresas estão sendo bem faladas ou criticadas. Essa informação pode ser muito útil para comprar ou vender ações, por exemplo. Nesse caso, a vantagem é a velocidade com que essas informações são obtidas [16].

Politica

Candidatos podem monitorar seus eleitores através de mídias sociais diferentes. Dessa forma, eles podem saber qual é a reação de seu eleitorado em relação a aspectos específicos das eleições, como por exemplo, o posicionamento das pessoas sobre suas propostas.

É possível usar um classificador de comentários em posts de redes sociais relacionados a política, poderíamos relacionar sentimentos a aspectos específicos que poderiam ser ou estar ligados aos próprios candidatos. Assim, saberíamos quem está sendo mais apoiado ou criticado.

Também é possível avaliar o sentimento das pessoas quanto a um assunto especifico, como “Aborto”, por exemplo. Dessa forma, cada candidato

poderia adaptar suas propostas e seu discurso de forma a agradar o maior número de pessoas [16] [17].

O capítulo a seguir apresenta métodos e técnicas para a construção de classificadores automáticos de sentimento, bem como algumas ferramentas utilizadas nesta tarefa.

3. Técnicas para Construção de Classificadores de Sentimento

Como já visto, o foco deste trabalho é na Análise de sentimento, que busca determinar a polaridade de textos opinativos. O objetivo principal é construir um processo de classificação de polaridade. Assim sendo, este capítulo vai expor alguns conceitos e técnicas relacionados à construção de classificadores de texto.

A seção 3.1 discute as principais abordagens para construção de classificadores. Antes de serem submetidos ao processo de classificação, os dados de entrada devem ser preparados (pré-processados), para ser possível classificá-los de forma automática. Como nosso objetivo é classificar comentários sobre produtos, veremos na seção 3.2 as técnicas de pré-processamento de texto.

A seção 3.3 apresenta algumas dos principais técnicas de AM supervisionada, que foi a abordagem adotada neste trabalho. Por fim, a seção 3.4 discute as dificuldades na classificação de texto para análise de sentimento.

3.1. Construção de classificadores de texto

Identificamos duas abordagens principais para construção de classificadores de texto: Aprendizagem de máquina e Engenharia do conhecimento (classificadores baseados em regras explícitas manualmente construídas).

A Abordagem baseada em conhecimento utiliza dicionários de termos com polaridade previamente determinada (em geral, adjetivos e advérbios) – seção 2.2.1. Já a Aprendizagem de Máquina (AM) necessita de textos etiquetados com as classes que se deseja utilizar na classificação dos comentários, e esses dados são utilizados para induzir o classificador através de um processo de aprendizagem automática. A AM tem sido a principal escolha na construção de classificadores de sentimento, devido às suas excelentes taxas de precisão e facilidade de uso.

3.1.1. Aprendizagem de máquina supervisionada e não supervisionada

A AM apresenta duas abordagens principais: supervisionada e não supervisionada. A AM supervisionada requer um corpus de dados previamente etiquetados com as classes desejadas. A seguir, esses textos e suas respectivas polaridades são usados para treinar os algoritmos de classificação.

A partir do momento em que um texto é tido como positivo, por exemplo, o algoritmo de aprendizagem extrai palavras e características que serão atribuídas à classe positiva. Porém, essas mesmas palavras (ou *features*) podem aparecer em textos pré-classificados como negativos. Nesse caso, podemos, por exemplo, levar em consideração a quantidade de vezes que cada palavra aparece em cada classe. Assim, as palavras que aparecem muitas vezes em documentos da classe positiva serão tidas como termos indicadores de polaridade positiva. Da mesma forma, a regra se aplica para palavras que se repetem na maioria dos textos de classe negativa. Dessa forma, esses termos podem ser usados para classificar novos textos [9].

A AM não supervisionada, diferentemente da supervisionada, não necessita de dados rotulados. Essa é uma clara vantagem sobre a aprendizagem supervisionada, já que esse modelo independe de um contexto especifico. Porém, essa abordagem é mais utilizada na criação de clusters (grupos) a partir dos dados de entrada. Nesses casos, podemos determinar a quantidade de clusters de saída (por exemplo, 2 – positivo e negativo). Mas não é possível indicar quais características são importantes para cada classe que desejamos como saída. Assim, o resultado de um algoritmo de cluster pode ser imprevisível. Por exemplo, o resultado do algoritmo de clustering pode devolver uma classe de comentários sobre celular iPhone e outra classe sobre Samsung. Assim, essas técnicas não são muito utilizadas para classificação de polaridade em análise de sentimentos.

3.1.2. Abordagem baseada em Léxicos Polarizados

A técnica mais usada dentro da abordagem baseada em regras explícitas utiliza léxicos polarizados pré-existentes (ex. SentiWordNet[2]). Nesse caso, cada palavra do léxico tem um valor quantitativo associado que indica a polaridade do

[2] https://sentiwordnet.isti.cnr.it/

termo. Esse valor pode variar de -1 a 1, de 1 a 10, ou qualquer outro intervalo pré-definido.

O conceito básico é que as notas mais próximas dos extremos devem ser consideradas positivas, ou negativas. Por exemplo, palavras entre -1 e 0 podem ser consideradas negativas (péssimo = -1; ruim = -0,5), e entre 0 e 1 são consideradas positivas (excelente = 1; bom = 0,5).

Cada palavra pode também receber um valor qualitativo, ou seja, ser classificada previamente entre positiva ou negativa [10] [11]. Assim, nessa abordagem, a polaridade de cada palavra é definida previamente e independe do contexto em que é aplicada. A polaridade da frase ou do texto completo é obtida somando-se os valores associados às palavras opinativas que ocorrem no trecho sendo analisado.

3.2. Técnicas de pré-processamento

Antes de classificar textos utilizando algoritmos de classificação de sentimento, geralmente é necessário pré-processar o documento, a fim de criar uma representação tratável computacionalmente. Em resumo, o pré-processamento recebe como entrada o texto original, e devolve sua representação simplificada, em forma de lista de palavras (aqui, as palavras ou termos compostos são considerados como atributos para classificação).

Em geral, o pré-processamento inclui uma etapa de redução do número de atributos considerados na classificação, removendo as palavras ou termos que não são importantes na análise de sentimento (conhecidas como stopwords). Também pode ser importante representar os atributos relevantes de formas mais eficiente, utilizando o seu radical (parte fixa da palavra, sem flexões), a fim de facilitar a identificação de termos com mesmo significado (processo de stemming).

Ou seja, o objetivo desse processo é selecionar os atributos mais importantes para representar cada texto, determinado também a forma melhor (palavra flexionada ou radical) para cada aplicação [20].

Antes de iniciar essas duas etapas descritas acima, é necessário dividir o texto de entrada em palavras isoladas, eliminando caracteres irrelevantes, como pontuação (etapa de Tokenização).

3.2.1. Tokenização

Essa etapa separa os termos do texto em unidades chamadas *tokens*. Um token pode não ser só uma palavra isolada, como também sinais de pontuação [21]. Esses elementos são separados por espaços. Espaços não são considerado tokens. No exemplo 5 identificamos alguns tokens em um comentário sobre um smartphone.

"Eu não gostei muito da tela desse celular"
Tokens: [Eu], [não], [gostei], [muito], [da], [tela], [desse], [celular]

Exemplo 5 - Identificação de tokens

A maneira como os tokens são separados pode variar, dependendo da língua sendo processada. Algumas línguas utilizam espaço como separador (ex., português, inglês), enquanto existem línguas que não possuem espaços entre as palavras [20]. Nesse caso, essa etapa depende da especificidade de cada idioma para fazer a identificação desses elementos.

3.2.2. Eliminação de Stopwords

Stopwords [22] são palavras que são consideradas irrelevantes para um texto a ser classificado. Geralmente, são elas: artigos, pronomes, preposições, advérbios, ou mesmo palavras muito frequentes no corpus de documentos considerado. Palavras que se repetem com frequência não são capazes de discriminar documentos de uma determinada classe. Assim sendo, sua remoção traz ganhos de eficiência para os algoritmos classificadores na grande maioria das vezes. A seguir, no exemplo 6 mostramos alguns exemplos de stopwords.

"As", "e", "os", "de", "para", "com", "sem", "foi", "da", "do", "uma", "acho", "que", "tem"

Exemplo 6 - Exemplos de stopwords

Mais stopwords podem ser obtidas através do gerenciador de pacotes NPM [23] para Javascript, que possui um dos maiores registros de palavras em diferentes idiomas no mundo. Podemos também definir uma lista stopwords personalizada através de algoritmos que calculam a frequência dessas palavras em um dado corpus de documentos, monitorando suas ocorrências em diferentes classes. Dessa forma, palavras que se repetem muito e que se distribuem homogeneamente entre diferentes classes podem ser consideradas stopwords e devem ser retiradas no pré-processamento.

3.2.3. Processo de Stemming

O *stem* de uma palavra representa seu radical, sem os prefixos e sufixos. O objetivo dessa etapa de processamento é representar variações de um mesmo termo com apenas um token. Por exemplo, o *stem* das palavras “amando”, “amado” e “amar” é “ama”. Todas essas palavras fazem referência ao verbo amar. Se não reduzíssemos essas palavras ao *stem*, cada um dos três termos seria considerado diferente. Assim, o classificador não conseguiria usar a ocorrência desses termos como evidência para classificar textos onde eles aparecem. A tabela 1 traz alguns exemplos de palavras e seus radicais.

Palavra	Stem
really	*realli*
experience/ experiment	*experi*
settle	*settl*
clarity / claritude	*clariti*
Easiest / easy	*eas*
Configure / configuration	*configur*

Tabela 1 - Stemming de palavras

Fonte: Resultados obtidos a partir do uso da biblioteca NLTK

O processo para reduzir palavras aos seus radicais já está automatizado e está disponível em algumas plataformas. Umas das mais conhecidas é a NLTK [24] para linguagem Python, e foi usada nesse projeto.

3.2.4. Uso de N-Grama

Um N-grama [25] é um elemento composto por 1 ou mais elementos, podendo ser visto como uma tupla de “n” termos. A representação por meio de N-gramas pode ser aplicada a palavras ou caracteres.

A ideia aqui é representar o texto de maneira a tentar manter o contexto o máximo possível. Por exemplo, no texto: “*O maior problema desse celular é o tamanho da tela*” podemos retirar N-gramas de nível 2, também chamados de bigramas, seriam os possíveis bigramas: (“O”, “maior”), (“maior”,” problema”), (“problema”, “desse”), (“desse”, “celular”), (“celular”, “é”), (“é”, “o”), (“o”,” tamanho”), (“tamanho”,” da”), (“da”,” tela”). Perceba que a representação desse texto em unigramas (N-grama de nível 1) são as próprias palavras.

Da mesma forma podemos representa-los como N-gramas de nível 3, os trigramas: (“O”, “maior”, “problema”), (“maior”, “problema”, “desse”), (“problema”, “desse”, “celular”), (“desse”, “celular”, “é”), (“celular”, “é”, “o”), (“é”, “o”, “tamanho”), (“o”, “tamanho”, “da”).

Isoladamente, palavras podem aparecer em diferentes textos com diferentes sentimentos. Por exemplo, a palavra “amor” pode aparecer em diferentes textos com diferentes sentimentos (veja o exemplo 7). Mas, um conjunto maior de palavras se repete com menos frequência, portanto tem maior probabilidade de manter o contexto original assim como o sentimento do texto (veja o exemplo 8).

*“Comprei esse celular por causa de sua **aparência**, não me arrependi” (positiva)*
*“Esse celular só tem **aparência**, o resto deixa a desejar” (Negativa)*

Exemplo 7 - Perca de contexto para a mesma palavra

*“Gostei desse celular porque **ele é bonito**” (Positiva)*
*“Uma das maiores qualidades desse celular é que **ele é bonito**” (Positiva)*

Exemplo 8 - Preservação do contexto para bigrama

Se retirarmos as stopwords, melhoramos ainda mais os resultados, veja no exemplo 9.

Textos com stopwords:

*"Gostei desse celular porque **é** **bonito**" (Positiva)*

*"Uma das maiores qualidades desse celular é que ele **é** **bonito**" (Positiva)*

Textos sem stopwords:

*"Gostei **celular bonito**" (Positiva)*

*"maiores qualidades **celular bonito**" (Positiva)*

Exemplo 9 - Preservação do contexto para bigrama sem stopwords

Agora, podemos retirar do texto um bigrama ("celular", "bonito"), que é mais representativo do que o trigrama ("ele", "é", "bonito"), já que, o primeiro pode ser uma referência a qualquer item que é bonito, mas, o segundo especifica o item elogiado.

A representação de textos usando N-gramas perde a relevância quando o "n" fica muito alto, isso porque quanto maior for o tamanho do N-grama mais difícil é de o mesmo se repetir em mais de um texto. Por exemplo, imaginemos um 7-grama. A probabilidade de outro documento apresentar o mesmo conjunto de 7 palavras é muito pequena, dessa forma, o processamento desse conjunto de palavras não apresenta melhoras no desempenho do classificador.

Além disso, um N-grama perde seu objetivo principal em manter o contexto quando os seus termos são divididos por pontuação, já que sinais de pontuação como vírgulas e pontos separam sentenças. Quando escolhemos usar N-gramas para representar nossos documentos, criamos não só os N-gramas de nível "N" escolhido, mas, todos os conjuntos de ordem "n" onde "n" é menor que "N" ($n < N$) e "n" maior que 0 ($n > 0$).

No exemplo 10 temos a representação do texto "*Eu comprei esse celular para minha namorada, ela adorou*" em formato de n-grama com $n = 3$.

N-grama com “n” igual a 3: (“Eu”, ”comprei”, ”esse”), (“comprei”, ” esse”,” celular”), (“esse”,” celular”,” para”), (“celular”,” para”,” minha”), (“para”,” minha”,” namorada”), (“minha”,” namorada”,” ela”), (“namorada”,” ela”,” adorou”).

N-grama com “n” igual a 2: (“Eu”, ”comprei”), (“comprei”, ”esse”), (” esse”, ”celular”), (” celular”, ”para”), (”para”, ”minha”), (” minha”,” namorada”), (” namorada”,” ela”), (” ela”,” adorou”).

N-grama com “n” igual a 1: (“Eu”), (“comprei”), (” esse”), (”celular”), (”para”), (”minha”), (”namorada”), (” ela”), (”adorou”).

Forma como o texto será representado: (“Eu”,” comprei”,” esse”), (“comprei”,” esse”,” celular”), (“esse”,” celular”,” para”), (“celular”,” para”,” minha”), (“para”,” minha”,” namorada”), (“minha”,” namorada”,” ela”), (“namorada”,” ela”,” adorou”), (“Eu”,” comprei”), (“comprei”,” esse”), (” esse”,” celular”), (” celular”,” para”), (” para”,” minha”), (” minha”,” namorada”), (” namorada”,” ela”), (” ela”,” adorou”), (“Eu”), (“comprei”), (” esse”), (”celular”), (”para”), (”minha”), (”namorada”), (” ela”),(”adorou”).

Exemplo 10 - Representação de documento em trigramas

3.3. Construção de Classificadores baseados em Aprendizagem Supervisionada

Como já visto anteriormente, a AM supervisionada é uma das abordagens que oferece melhores resultados na construção de classificadores. Assim, essa foi a abordagem adotada neste trabalho. Esta seção apresenta alguns dos algoritmos mais utilizados na construção de classificadores de texto: Naive Bayes, SVM e Random Forest.

Um dos maiores problemas na classificação da polaridade de comentários é a existência de palavras de contexto negativo em documentos positivos e vice e versa (ex., “não”, “nunca”, “sim”). Além disso, é comum encontrar em comentários ao mesmo tempo elogios e críticas, o que dificulta a classificação, já que o algoritmo precisa calcular a intensidade de cada sentimento expresso para obter um média. Cada algoritmo aborda esses problemas de formas diferentes.

3.3.1. Naive Bayes

Naive Bayes [26] é um método de construir classificadores que analisa termos e documentos a partir do princípio proposto pelo teorema de bayes [37]. Esse teorema calcula a probabilidade de ocorrência de um evento baseado em conhecimento sobre ele. No contexto de análise de sentimentos, o classificador *Naive Bayes* classifica um documento a partir dos termos presentes no mesmo.

Por exemplo, se um documento é composto por termos considerados positivos, será classificado como positivo; seguindo o mesmo princípio para termos e documentos negativos. Contudo, não é tão simples classificar um termo em positivo ou negativo, isso porque um mesmo termo pode aparecer em documentos de ambas as classes. Dessa forma, o algoritmo em questão calcula a probabilidade condicional de um termo pertencer a uma classe específica baseado na quantidade de vezes que esse termo aparece em documentos dessa classe.

O classificador *Naive Bayes* assume independência entre as *features* (palavras, ou grupos de palavras quando usamos N-gramas). No caso de um texto, ele assume que cada palavra não tem relação com as outras, dessa forma basta saber qual é a probabilidade da *feature* presente em um documento pertencer a alguma classe para que possamos calcular a polaridade desse texto.

O algoritmo começa seu processamento criando uma tabela a partir dos textos classificados previamente. Nessa tabela, chamada de tabela de frequência, é guardada cada palavra e a quantidade de vezes que ela aparece em documentos de diferentes classes. Vejam o exemplo 11.

Com StopWords:

"Não gostei da tela desse celular" (1) (negativa)

"A bateria desse celular dura bastante" (2) (positiva)

"Esse celular é bom" (3) (positiva)

Sem StopWords:

"Não gostei tela celular" (1) (negativa)

"bateria celular dura bastante" (2) (positiva)

"celular bom" (3) (positiva)

Exemplo 11 - Detecção de features em sentenças

Podemos observar que as palavras "**não**", "**gostei**" e "**tela**" só aparecem em documentos negativos, logo, será atribuída a elas uma alta probabilidade de serem negativas. O mesmo se aplica às palavras **"bateria", "dura", "bastante"** e **"bom"** para as classes positivas**.** Já o termo **"celular"** aparece nas duas classes, entretanto, aparecendo com mais frequência em documentos da classe positiva. Logo, terá maior probabilidade de representar a classe positiva caso seja encontrada em um documento ainda não classificado.

Podemos calcular a probabilidade de uma palavra qualquer pertencer a uma classe especifica através da fórmula da equação 1.

$$P("classe"|\ "feature") = \frac{P("feature"|\ "classe") * P\ ("classe")}{P("feature")}$$

Equação 1 - Cálculo da probabilidade de uma feature pertencer a uma classe

Para classificar um documento, o algoritmo calcula a probabilidade de cada *feature* de pertencer à classe positiva e a probabilidade de pertencer à classe negativa. Com a probabilidade condicional de cada *feature* em mãos, as probabilidades são combinadas para que possamos classificar o documento. Em resumo, classificamos cada documento com base na probabilidade condicional de cada termo presente nele.

São comuns trabalhos no meio acadêmico que usam o algoritmo Naive Bayes para análise de sentimentos. Esse algoritmo é comumente usado desde a análise em redes sociais [27] a análises de comentários em sites comerciais [28].

3.3.2. Suport Vector Machine (SVM)

O objetivo do classificador SVM é a criação de um hiperplano que separe linearmente diferentes classes de maneira mais precisa possível. Os hiperplanos são determinados por um subconjunto de exemplos de treinamento. A partir de uma entrada pré-classificada, o algoritmo cria um modelo onde cada ponto faz referência a um documento classificado. A imagem 1(a) mostra a criação de um hiperplano para um conjunto de dados.

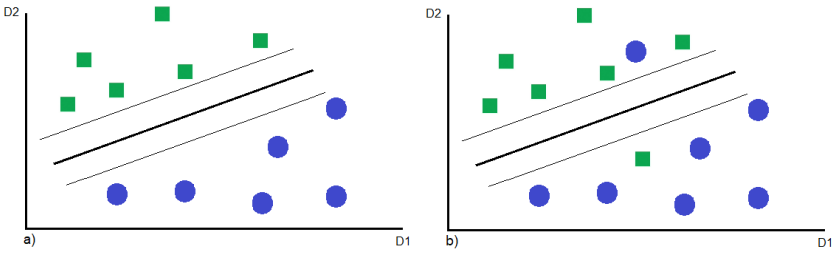

Imagem 1 - (a) Espaço linearmente separável, (b) Espaço não linearmente separável

Fonte: autoria própria

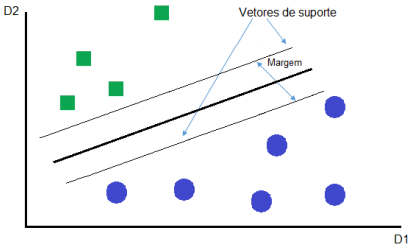

Imagem 2 - Vetores de suporte e margem de separação

Fonte: autoria própria

O hiperplano pode ser construído com precisão para dados linearmente separáveis (Imagem 1a). O objetivo é conseguir as maiores margens possíveis (imagem 2). Para casos onde os dados não são linearmente separáveis (imagem 1b), precisamos usar técnicas para separar esses conjuntos de dados

linearmente. Nesse caso, utiliza-se o teorema de Cover [29], que cria um novo plano com mais dimensões a partir do plano original. O teorema de Cover diz que é possível transformar um espaço de dados não linearmente separáveis em um espaço de dados linearmente separáveis com alta probabilidade através da aplicação de transformações não lineares. Na imagem 3 temos um plano após essas transformações. O hiperplano da imagem 3a se transforma no hiperplano da imagem 3b.

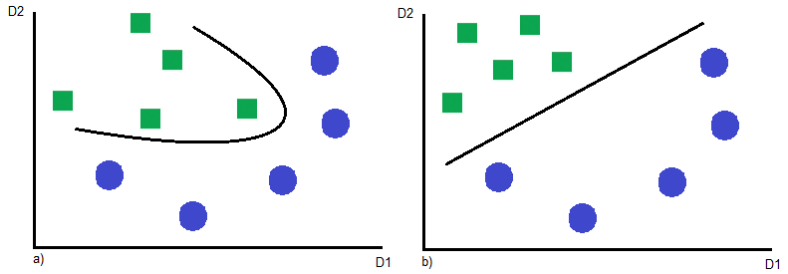

Imagem 3 - Hiperplano após transformação linear

Fonte: autoria própria

Em alguns casos, dependo do conjunto de dados usados no treinamento, essas transformações podem se tornar computacionalmente custosas caso o número de dimensões cresça muito. Para contornar esse problema, usamos simplificações que são aplicadas por meio de funções *Kernel*. A imagem 4 mostra um hiperplano depois da aplicação de uma função *Kernel*.

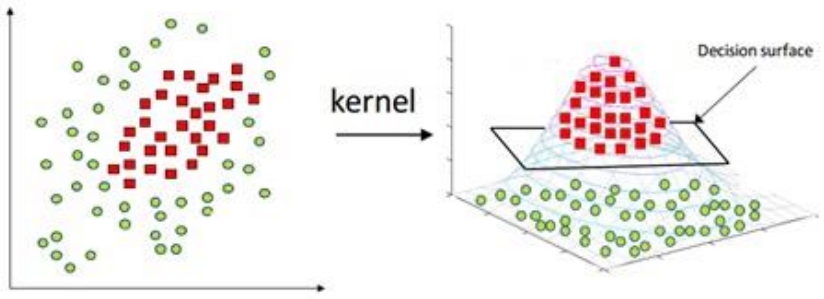

Imagem 4 - Aplicação de função Kernel

Fonte: Newtechdojo, Learn Support Vector Machine, december 28, 2017

Funções *Kernel* realizam transformação de espaço de maneira pouco custosa. Uma vez criado um hiperplano mais preciso possível, o classificador calcula novos pontos a partir de novas instâncias (documentos) e tentar localizar

esses pontos em alguma das regiões do hiperplano, onde cada região será uma classe diferente. À medida que o SVM tenta classificar novos documentos, pode acabar classificando documentos de forma errônea. Quando isso acontece, o modelo é atualizado para que o novo plano englobe corretamente o último ponto calculado, usando as funções de transformação já comentadas acima.

Um dos problemas que essas correções podem acarretar é o chamado *overfitting (*“super ajustamento”), que ocorre quando o modelo está tão bem ajustado aos dados já processados que perde sua capacidade de classificar novos dados. Nesse caso, é usada uma constante de erro “C” para decidir a intensidade do ajuste após um erro ser encontrado. SVM também é um algoritmo popular em análise de sentimento e é conhecido principalmente por apresentar resultados promissores [30].

3.3.3. Random Forest

Random Forest [31] é um algoritmo de aprendizagem de máquina baseado em árvore de decisão. Uma árvore de decisão classifica uma instância baseada em variáveis decisórias seguindo o caminho da raiz até as folhas (Imagem 5). Variáveis decisórias (ou atributos) são usadas para tomar decisões, e suas respostas formam caminhos específicos em uma árvore. Por exemplo, a imagem 5 mostra a árvore de decisão criada para prever se haverá jogo de futebol em um clube se base em 3 atributos: se faz sol ou não, se a temperatura é maior que 35º (graus) ou não, e se é feriado ou não.

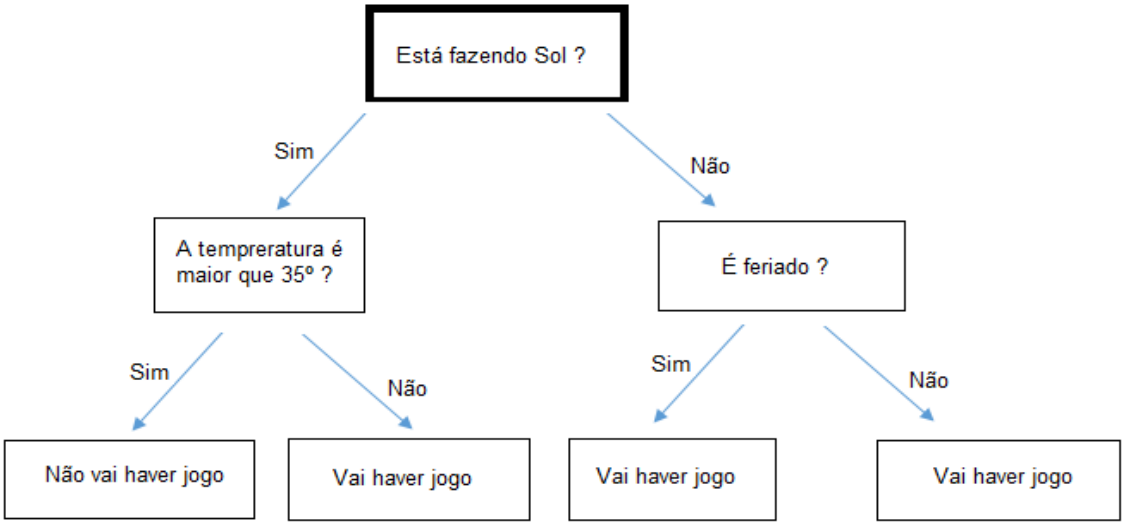

Imagem 5 - Árvore de decisão para jogo de futebol

Fonte: autoria própria

Para o contexto de análise de sentimento, cada atributo seria uma *feature*, e cada resposta seria uma possível característica dessa *feature* [32]. Uma árvore de decisão então poderia ser construída a partir de um subconjunto de dados já classificados.

Infelizmente, essa abordagem oferece algumas desvantagens. Como já citado aqui, sabemos que a linguagem natural pode variar muito, e que uma mesma palavra que aparece em classes positivas também pode aparecer em classes negativas. Então, construir apenas uma árvore não é o suficiente para a criação de um modelo eficiente. Assim sendo, o algoritmo *Random Forest* propõe a criação de várias árvores de decisão baseadas em subconjuntos aleatórios de uma base de dados. Dessa forma, podemos classificar um documento baseado não só em uma árvore, mas, sim em uma “floresta” (Imagem 6).

O *overfitting* também pode acontecer para essa técnica. Nesse caso, o que acontece é que uma árvore de decisão pode ficar tão profunda, ou seja, com tantos atributos, que nenhuma instância de teste possa ser corretamente classificada. Para esse problema, a solução é “podar” a árvore em determinados pontos, aumentando a generalidade de sua decisão. Pode parecer que podar uma árvore de decisão é jogar fora informações importantes e diminuir sua precisão, mas esse é um problema estatístico em que a generalização é preferida (até um certo ponto, claro) para que possamos maximizar nossos resultados.

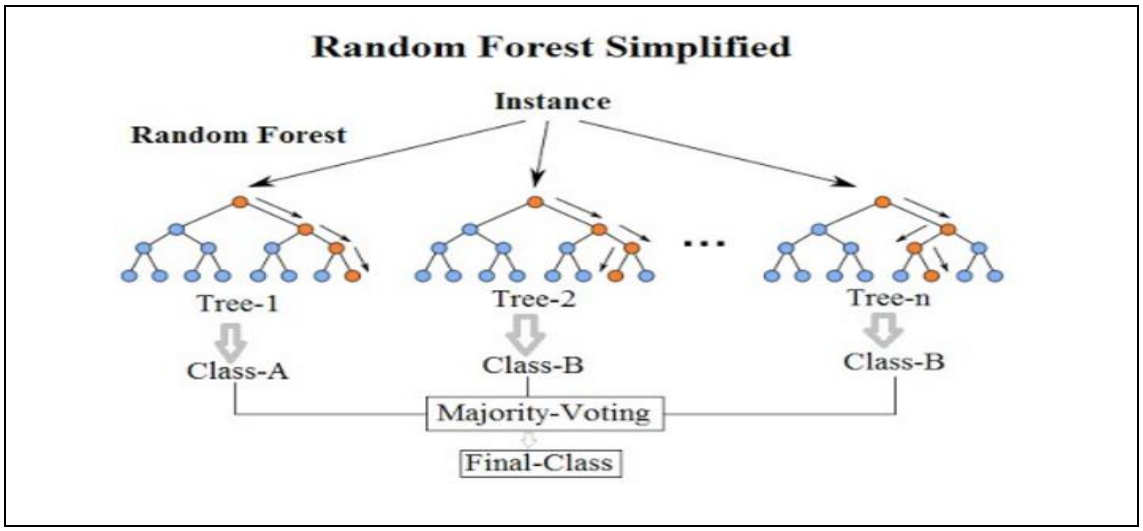

Imagem 6 - Random Forest

Fonte: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

3.3.4. Métricas de avaliação

Para avaliar o desempenho de um classificador, ou seja, avaliar sua competência em classificar instâncias, usaremos algumas métricas de avaliação tradicionais. São elas: False Positive, False Negative, True Positive e True Negative. Utilizamos ainda mais algumas métricas que são derivadas dessas quatro primeiras, são elas: precisão, cobertura, F-Measure e AUC.

● **False Positive (FP)**: É o número de vezes que uma classe foi avaliada como positiva de forma incorreta (era negativa).

● **False Negative (FN)**: É o número de vezes que uma classe foi avaliada como negativa de forma incorreta (era positiva).

● **True Positive (TP)**: É o número de vezes que uma classe foi avaliada como positiva de forma correta (era positiva).

● **True Negative (TN)**: É o número de vezes que uma classe foi avaliada como negativa de forma correta (era negativa).

3.3.4.1. Precisão

Precisão é uma medida popular para avaliar o resultado de classificadores. A precisão do classificador para a classe positiva é a razão do número de vezes em que um documento positivo foi predito corretamente para a classe positiva pelo número de documentos preditos como sendo da classe positiva. O mesmo ocorre para a classe negativa. Ou seja, a precisão define o quão eficiente é o classificador em classificar uma classe específica.

A precisão pode ser calculada para todas as classes sendo consideradas. Na Equação 2, temos a fórmula para o cálculo da precisão para documentos da classe positiva. TP indica o número de vezes que um documento foi corretamente classificado como positivo, e FP indica o número de vezes em que um documento da classe negativa foi classificado como positivo. A equação para a classe negativa é idêntica, apenas trocando TP por TN e FP por FN.

$$Precisão = \frac{TP}{TP + FP}$$

Equação 2 - Cálculo da precisão para a classe positiva

3.3.4.2. Cobertura

Cobertura, ou *Recall,* é a razão entre o número de vezes que a classe foi predita corretamente pelo número real de documentos na base que pertencem a essa classe. Na Equação 3, temos a fórmula para o cálculo da cobertura.

Assim como a precisão, TP se refere ao número de documentos positivos corretamente classificados como positivos, e FN se refere ao número de vezes que o classificador classificou um documento positivo como sendo negativo. A cobertura pode ser calculada para classes positivas e classes negativas.

$$cobertura = \frac{TP}{TP + FN}$$

Equação 3 - Cálculo da cobertura

3.3.4.3. Acurácia

A acurácia, diferentemente da precisão e da cobertura, não mede o desempenho do classificador para uma classe especifica, mas sim para a combinação das duas classes. Ou, seja, mede a exatidão do classificador. Na Equação 4 temos a fórmula para o cálculo da Acurácia.

$$Acuracia = \frac{TP + TN}{TP + TN + FP + FN}$$

Equação 4 - Cálculo da Acurácia

3.3.4.4. F-Measure

F-Measure ou *F1 Score* é uma métrica derivada da precisão e da cobertura, sendo a média harmônica entre essas medidas. Essa medida só cresce quando ambas as outras medidas crescem. Assim, podemos dizer que o classificador chega ao seu melhor resultado quando a F-Measure atinge seu ponto máximo. Na Equação 5, temos a fórmula para o cálculo da F-Measure.

$$Fmeasure = \frac{2 * Precisão * Recall}{Precisão + Recall}$$

Equação 5 - Cálculo da F-Measure

3.3.4.5. AUC - Area Under the ROC Curve

Essa métrica representa a medida da área embaixo de uma curva formada pelo gráfico feito com taxas de TP (verdadeiros positivos) e FP (falsos positivos) (imagem 7). A principal vantagem dessa métrica é que ela mede o desempenho do classificador em vários pontos de corte diferentes.

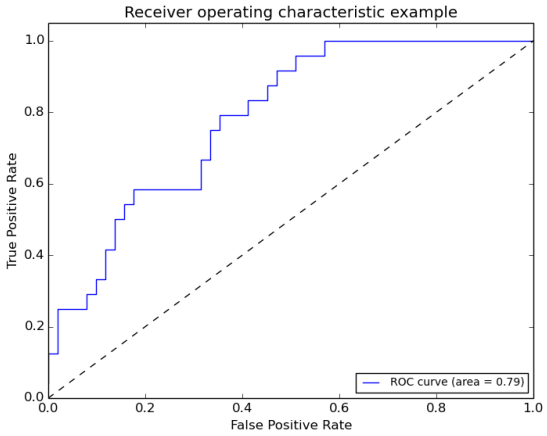

Imagem 7 - Gráfico da área sobre a curva

Fonte: https://scikit-learn.org/stable/

A métrica AUC é geralmente vista como uma métrica de ranqueamento, ou seja, quando seus valores são altos significa que o classificador tem boa capacidade de ranquear os documentos classificados. Além disso, a métrica AUC representa a capacidade de um classificador de escolher corretamente uma classe, ou seja, sua capacidade de classificar como positivo o que é positivo e como negativo o que é negativo.

3.4. Ferramentas e Bibliotecas de Apoio

Esta seção tem por objetivo apresentar algumas ferramentas e bibliotecas que podem ser usadas como apoio ao desenvolvimento de classificadores. Mostraremos aqui as ferramentas utilizadas no desenvolvimento deste trabalho:

WEKA e NLTK. Por fim, faremos uma breve apresentação da linguagem de programação PHYTON.

3.4.1. WEKA

A ferramenta WEKA [33] disponibiliza um conjunto de algoritmos baseados em aprendizagem de máquina para criar e testar algoritmos de classificação, contando com as técnicas apresentadas acima e outras que não foram incluídas aqui.

A ferramenta recebe como entrada um conjunto de treinamento com as classes etiquetadas, para a partir de aí induzir os classificadores. Os dados de treinamento são representados em um arquivo no formato arff, que relaciona as instâncias classificadas com suas *features* (atributos). Na imagem 8, temos um exemplo de arquivo de entrada representando atributos e instâncias para um problema de classificação que visa decidir se vai ou não haver jogo de acordo com um conjunto de atributos.

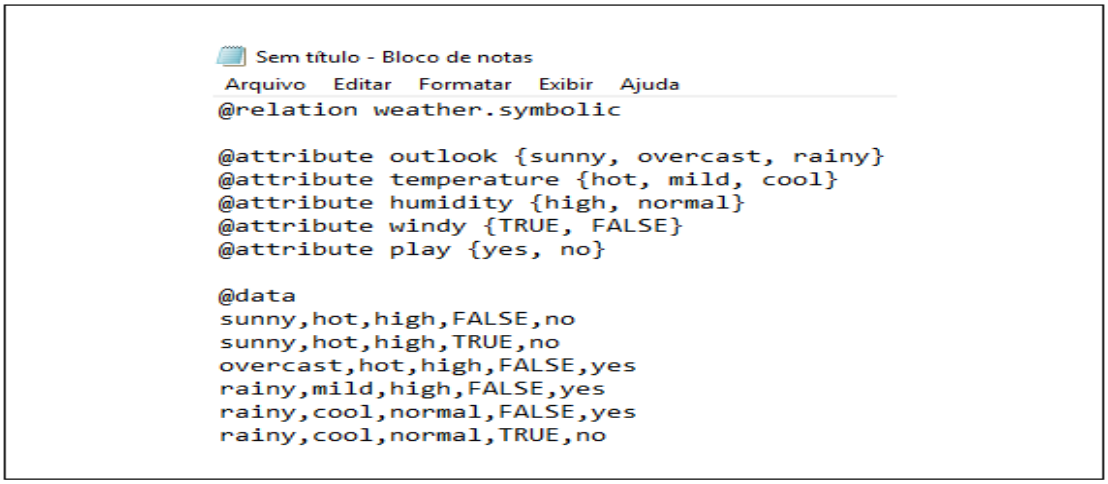

Imagem 8 - Exemplo de arquivo “.arff”

A partir do momento que o arquivo é dado como entrada, o próximo passo é escolher o algoritmo que será utilizado (Imagem 9, circulado em vermelho) para indução do classificador com aquele conjunto de dados.

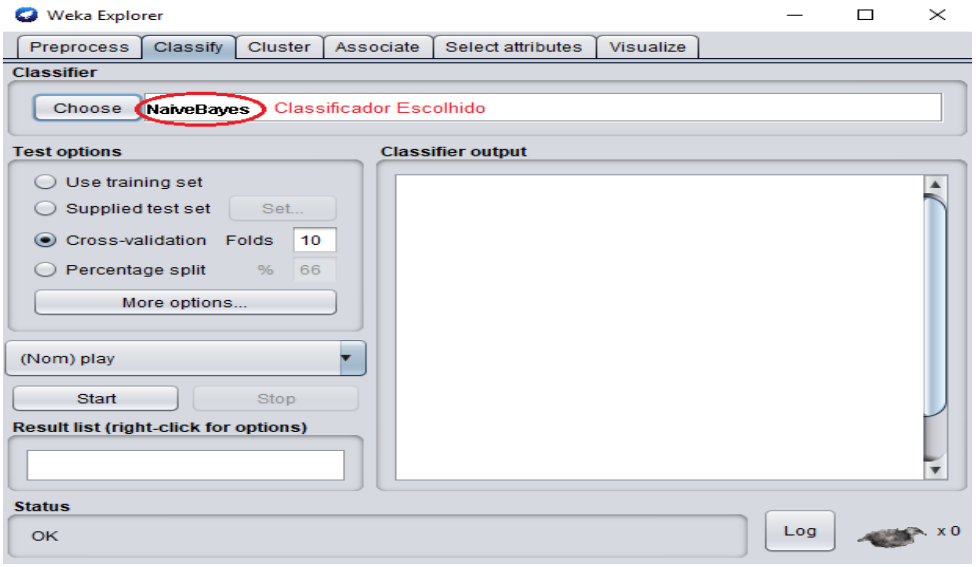

Imagem 9 - Escolhendo um classificador

O conjunto de treinamento pode ser dividido automaticamente pela ferramenta, para também testar o desempenho do classificador induzido. Esse procedimento é conhecido como *holdout*. Ao final do processamento, WEKA apresenta os resultados obtidos por um classificador (Imagem 10).

```
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall   F-Measure   MCC      ROC Area  PRC Area  Class
                 1,000     1,000     0,600       1,000    0,750       ?        0,333     0,589     yes
                 0,000     0,000     ?           0,000    ?           ?        0,333     0,450     no
Weighted Avg.    0,600     0,600     ?           0,600    ?           ?        0,333     0,533
```

Imagem 10 - Resultados da classificação WEKA

3.4.2. Biblioteca NLTK

NLTK [24] [34] é uma biblioteca/plataforma para linguagem PYTHON que apresenta um dos maiores conjuntos de corpora e dados em várias línguas. Entre as funcionalidades dessa biblioteca utilizadas nesse projeto, estão:

- **Tokenização**: transforma um texto em uma lista de palavras, facilitando a manipulação de cada *feature*.

- **Tagging**: Realiza etiquetagem da classe gramatical das palavras nas frases do documento de entrada (adjetivo, adverbio, verbo, substantivo, etc).

- **Lista de Stopwords**: Possui lista de palavras consideradas irrelevantes, permitindo assim que possamos identificar e retirar essas palavras de nossos textos.

● **Stemming**: reduz cada palavra ao seu radical (stem).

● **N-Gramas:** rearranja termos da lista de palavras isoladas em tuplas de ordem N.

3.4.3. Linguagem PYTHON

PYTHON[3] é uma linguagem de alto nível criada por Guido van Rossum em 1991. Entre seus principais objetivos estão a produtividade e a legibilidade de código. Para isso, ela adota uma série de características especiais [35] que a diferenciam de outras linguagens, por torná-la mais prática.

Essa linguagem foi escolhida para implementação do projeto por suportar diferentes estruturas (tuplas, listas, dicionários), além de facilitar a manipulação dessas estruturas através de funções pré-definidas. Além disso, PYTHON é geralmente utilizada por diversas APIs, e é base para diversas bibliotecas, como a NLTK, que é utilizada nesse projeto.

3.5. Desafios e limitações

Um dos principais desafios da classificação da polaridade de textos na Análise de sentimentos é a especificidade de cada linguagem. Outros desafios são as formas enganosas de expressar sentimento, como ironia e sarcasmo, por exemplo.

Além disso, também é constante a presença de *bots* e *trolls* nas mídias socias, fazendo com que os classificadores sejam treinados com informações falsas, ou seja, informações que não vieram de uma pessoa ou pelo menos não de forma honesta e espontânea. Já existem técnicas para detectar expressões enganosas [18] e entidades maliciosas [19].

[3] https://www.python.org/

4. Trabalho Realizado

Na seção 4.1 apresentamos uma visão geral da arquitetura de nosso projeto, as etapas de pré-processamento, bem como o corpus escolhido para a classificação (seção 4.2.). Detalhamos também as técnicas de pré-processamento utilizadas não só para representação mais eficiente de cada documento durante o processo de tratamento quanto para o processo de criação do arquivo de saída arff. Na seção 4.3 explicitamos como o nosso algoritmo de pré-processamento implementa tais técnicas e na seção 4.4 falamos como a ferramenta WEKA classifica cada instancia de teste.

4.1 Visão geral

Nosso trabalho criou uma base de dados (corpus) classificada manualmente. Com essa base de dados, a etapa de pré-processamento trata o conteúdo de cada texto aplicando as técnicas de pré-processamento desejadas. Ao final do processamento um arquivo arff é gerado para servir como entrada da Ferramenta WEKA.

Através da interface da WEKA podemos escolher qual classificador usar e como dividiremos nossas instâncias de entrada em treinamento e testes. A ferramenta treina o classificador escolhido, testando-o através da técnica de *holdout*, gerando assim os resultados que usaremos para avaliar sua performance. Na imagem 11 apresentamos a arquitetura de nosso projeto.

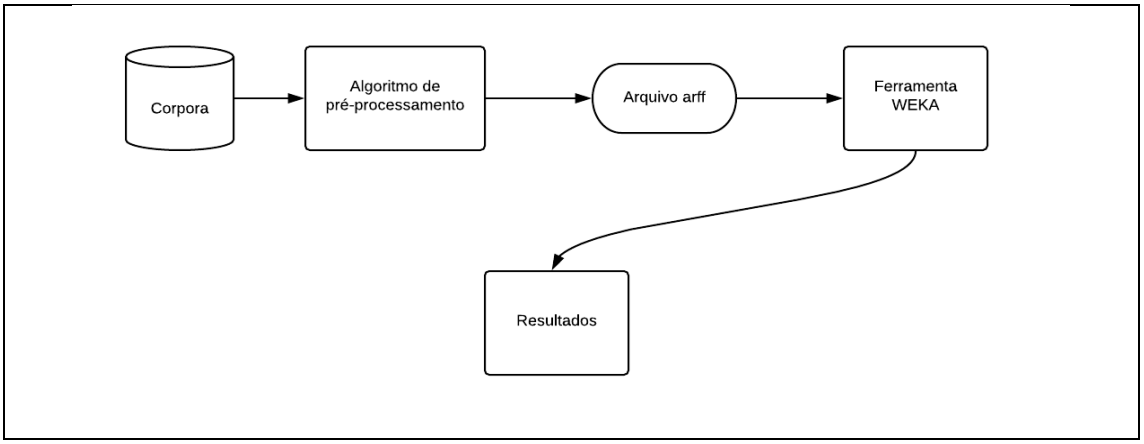

Imagem 11 - Arquitetura geral do projeto

4.2. Corpus de Comentários

O corpus de documentos coletado para este projeto consiste em comentários sobre smartphones. Os comentários em inglês foram retirados do site da *Amazon.com* [36]. Foram 300 comentários coletados. Cada texto foi escrito pelos usuários em um campo de livre preenchimento, ou seja, o autor do comentário teve liberdade para escrever o que quiser sobre o produto.

Cada documento representa um comentário e o tamanho de cada documento variou de 1 a 3 KB. Os comentários foram escolhidos manualmente e apresentam-se distribuídos igualmente entre as classes positivas e negativas para maximizar a capacidade de aprendizagem do classificador. Abaixo na imagem 12 amostras de nossa base de documentos.

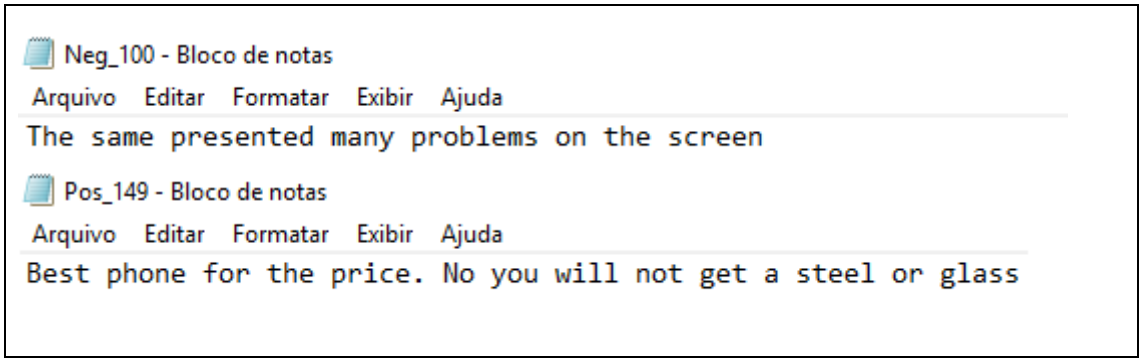

Imagem 12 - Exemplos da base de documentos

Os documentos coletados depois de pré-processados são usados como entrada para o ciclo de treinamento e testes com a ferramenta WEKA. Cada documento é carregado pelo algoritmo de pré-processamento em uma lista de textos (cada comentário é um elemento da lista). A leitura dos documentos é feita com a função “open” e ”.read()” da linguagem PYTHON.

Cada lista será uma lista de *strings* (maneira como textos são representados em linguagens de programação). Cada documento que será lido recebe um nome padronizado. Para o caso de documentos positivos serão nomeados para “pos_X.txt”, onde X é um número de 1 a 150, assim como os negativos seguem o mesmo padrão para o nome “neg_x.txt”.

4.3. Fase de pré-processamento

Na fase de pré-processamento. Nosso objetivo é tratar o texto usando variações das técnicas já mencionadas nesse documento, além de representar

o texto de entrada em um formato válido para que possa ser utilizado como entrada para a ferramenta WEKA.

Nessa fase, existem técnicas que serão aplicadas para facilitar o tratamento do texto e para simplificar sua representação posterior no arquivo de saída. Algumas técnicas são consideradas obrigatórias. Existem também técnicas opcionais e nesse caso, pudemos escolher usa-las ou não em cada experimento dependendo de nosso objetivo.

A imagem 13 mostra um fluxograma explicitando o funcionamento de nosso algoritmo de pré-processamento. Vale a pena observar mais uma vez que quando a técnica N-grama não é escolhida para ser aplicada no algoritmo de tratamento, dizemos que usamos N-grama para N = 1.

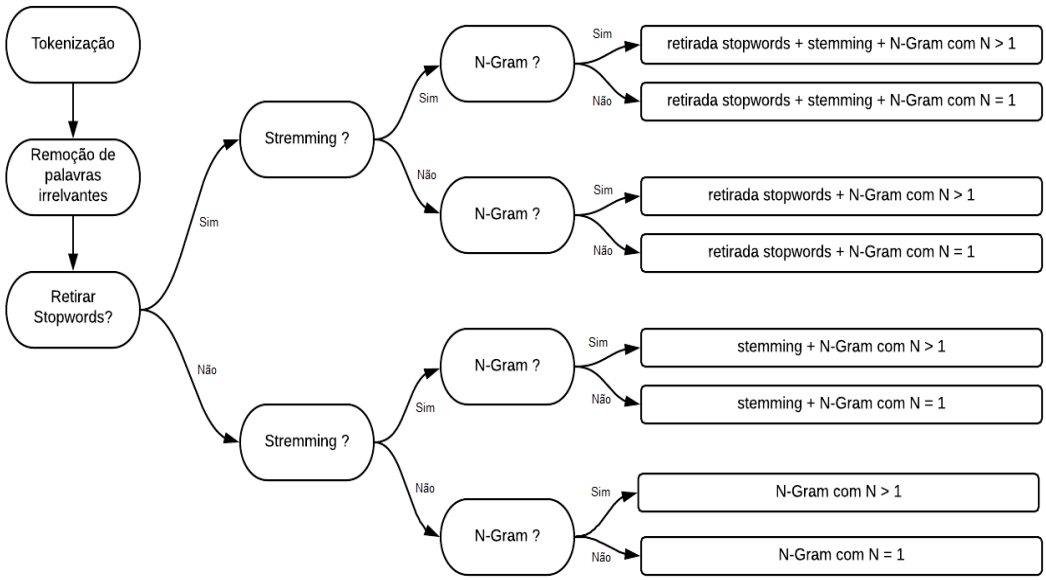

Imagem 13 - Fluxograma para algoritmo de pré-processamento

Fonte: Criado com a ferramenta disponível no site https://www.lucidchart.com

4.3.1. Técnicas Obrigatórias

Algumas técnicas são aplicadas obrigatoriamente no texto, visando criar uma representação do texto em forma de lista de *tokens*, que seja tratável pelos algoritmos de pré-processamento. A partir dessa lista serão escolhidos os termos para representar cada documento, excluindo as *features* (palavras) sem nenhum potencial para análise de sentimento. Além disso, é necessário representar cada

documento em um formato padrão para podermos utiliza-lo na próxima etapa (classificação com o WEKA).

4.3.1.1. Tokenização dos comentários

Inicialmente cada documento de entrada é tokenizado. Isso significa que cada texto será agora representando como uma lista de palavras (nossas *features*). Além disso, colocamos todas as palavras em letras minúsculas, usando a função ".lower()" da linguagem PYTHON. Essa transformação facilita a identificação dessas palavras no futuro, já que para uma linguagem de programação a palavra "AMOR" é diferente da palavra "amor", mesmo que para nosso propósito essas palavras sejam iguais. A tokenização é realizada através da função "word_tokenize" da biblioteca "nltk", ver imagem 14.

```
from nltk.tokenize import sent_tokenize, word_tokenize
.....
commentsTokenized = [];
for comment in comments:
        commentsTokenized.append(word_tokenize(comment.lower()))
```

Imagem 14 - Processo de tokenização

4.3.1.2. Detecção e remoção de palavras e caracteres desnecessários

No próximo passo, retiramos palavras que consideramos irrelevantes para alcançar bons resultados. Diferente das stopwords, essas palavras não são predefinidas por bibliotecas e não precisam necessariamente pertencer a uma classe gramatical. Primeiramente, substituímos cada caractere de uma palavra que não é um número ou letra do alfabeto, o objetivo aqui é representar essas palavras de forma mais simples além de facilitar a remoção desse termo posteriormente por estar malformado. Veja a Tabela 2 para exemplos dessas transformações. Logo após, descartamos as palavras que depois da substituição de caracteres são consideradas irrelevantes. Essa decisão é baseada em duas (2) regras:

● Número de caracteres menor que 2: depois da substituição de caracteres inválidos (que não são letra ou números), descartamos todas as palavras

menores que 2 caracteres, palavras com 1 ou 0 caracteres (essas ultimas sendo palavras vazias)

● Números: Palavra que possuem números em sua composição são mantidas, mas, palavras que são compostas inteiramente por números são descartadas.

Veja a tabela 2 para exemplos de palavras descartadas.

PALAVRA	Após transformação	Descartada?
D0g	D0g	NÂO
I7	I7	NÂO
Don’t	dont	NÂO
‘e	e	SIM
820	820	SIM
“”	*empty*	SIM
()	*empty*	SIM

Tabela 2 - Retirada de caracteres e palavras irrelevantes

A transformação e retirada dessas palavras foi realizada através de uma função (imagem 15) que analisa as características baseada nas regras já descritas acima. O objetivo dessa etapa é remover palavras que são incapazes de expressar sentimentos.

```
.....
for comment in comments:
        commentFiltered = [];
        for word in comment:
                newWord = replaceInvalidCaracteres(word)
                if not verifyUselessWord(newWord):
                        commentFiltered.append(newWord),
        commentsFiltered.append(commentFiltered);
.....
def isNumber (word):
        for caracter in word:
                if not ('0'<= caracter <= '9'):
                        return False;
        return True;
```

```
.....
def replaceInvalidCaracteres(word):
        newString = re.sub('[^a-z0-9]', '', word)
        return newString;
.....
def verifyUselessWord(word):
        if len(word) < 2:
                return True;
        if isNumber(word):
                return True;
        return False;
```

Imagem 15 - Retirada de caracteres e palavras irrelevantes

4.3.1.3 Representação das features em tuplas

Após a aplicação de técnicas opcionais de pré-processamento, optamos por representar cada feature em formato de tupla. Isso se deve ao fato de que para representação de features com a técnica N-grama, o formato de tupla é obrigatório durante o processamento. Dependendo do valor de “N”, podemos ter uma saída composta por tuplas de até “N” termos. Contudo, antes de gravar essas features em um arquivo arff, representamos cada tupla como seus elementos separados pelo caractere underscore (“_”). Veja o exemplo 12.

Texto original: “comprei um celular”
Formato de tupla para 3-Gram: (“comprei”, “um”, “celular”), (“comprei”, “um”), (“um”, “celular”), (“comprei”), (“um”), (“celular”)
Criação de arquivo de saída: “comprei_um_celular”, “comprei_um”, “um_celular”, “comprei”, “um”, “celular”

Exemplo 12 - Representação de tuplas no arquivo saída

Esse tratamento é realizado porque a ferramenta WEKA tem limitações a respeitos dos caracteres presentes em seus arquivos de entrada. O processo de representação das features em tuplas é realizado ao chamarmos a função de N-grama na biblioteca NLTK.

4.3.1.4 Criação da estrutura Bag-of-words

Essa é uma das etapas finais de processamento do texto antes da geração o arquivo de saída. Aqui criamos uma coleção de palavras que estão presentes em todos os comentários processados, que pode ser vista como a união de cada documento positivo e negativo a nível de palavra em uma só estrutura. Essa estrutura é usada para a criação das instâncias de treinamento e teste que serão representadas como uma bag-of-words. Bag-of-words é uma maneira simplificada de representar documentos, nessa estrutura, cada palavra é representada de maneira indireta através de um valor “0” ou “1”. 0: não está presente, 1: está presente (Vê imagem 16 para exemplo de Bag-of-words). A imagem 17 mostra a implementação da criação dessas estruturas em nosso algoritmo de pré-processamento.

Imagem 16 - Representação bag-of-words

Fonte:https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6

```
bagOfWords = []
for comment in Arquivos1:
        for tupla in comment:
                if tupla not in bagOfWords:
                        bagOfWords.append(tupla);

for comment in Arquivos2:
        for tupla in comment:
                if tupla not in bagOfWords:
                        bagOfWords.append(tupla);
```

Imagem 17 - Criação da estrutura bag-of-words

4.3.1.5 Geração do arquivo de saída

Essa é a última etapa do processo. O objetivo agora é criar um arquivo de saída arff para uso posterior na ferramenta WEKA. Para a criação desse arquivo, usamos todos os documentos já processados representados como um bag-of-words. O arquivo que será usado de entrada para a ferramenta de classificação requer um formato especifico representado na imagem 18.

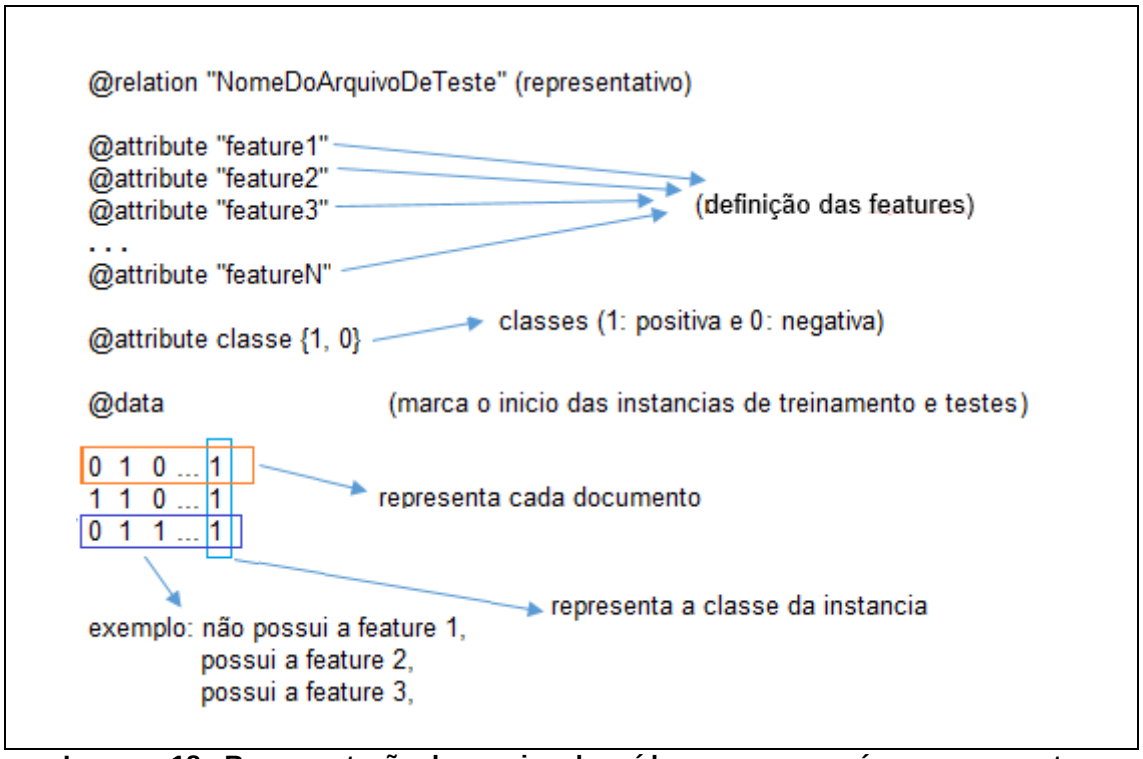

Imagem 18 - Representação do arquivo de saída para nosso pré-processamento

Decidimos representar a classe negativa por “0” e classe positiva por “1”. Na imagem 19c temos o preenchimento do arquivo de saída com os atributos e na imagem 19a a criação das instâncias de treinamento e testes.

```
.....
a)
for tupla in bagOfWords:
        if tupla in comment:
                arquivoSaida.write("1 ")
        else:
                arquivoSaida.write("0 ")

if typeDoc == POS_DOCUMENT:
        arquivoSaida.write("1")
else:
        arquivoSaida.write("0")
```

```
.....
b) def unifyTupla(tupla):
        newElement = tupla[0]
        for element in tupla[1:]:
                newElement += "_" + element;
        return newElement;
```

```
.....
c) for tupla in bagOfWords:
                arquivoSaida.write("@attribute " + unifyTupla(tupla) + " {0,1}\n");
```

Imagem 19 - a) Criação das instâncias de treinamento, b) Mudando a forma de representar as features, c) Definindo os atributos no arquivo arff,

4.3.2. Técnicas Opcionais

Algumas das técnicas empregadas nos documentos são opcionais e podem ser controladas por variáveis de controle do projeto. O algoritmo foi construído dessa forma para que possamos testar variações diferentes de técnicas de pré-processamento de texto

4.3.2.1 Retirada de StopWords

Uma das técnicas que podem ser empregadas no processamento dos comentários é a retirada de stopwords. Diferente da retirada das palavras desnecessárias da seção **4.3.1.2** desse documento, as stopwords estão definidas em listas que podem ser obtidas em bibliotecas virtuais. No nosso caso, usamos a lista fornecida pela biblioteca NLTK. Entretanto, como decisão de projeto, escolhemos retirar algumas dessas palavras por considerarmos relevantes ao nosso contexto de análise de sentimento.

As palavras que não foram retiradas são os advérbios, filtramos tal classe através da funcionalidade *tagging* da biblioteca NLTK. Advérbios são muito importantes para classificação de texto, já que esses termos podem intensificar ou até inverter o valor da polaridade de uma *feature*. Tendo em vista que nossos classificadores assumem independência entre as palavras, os advérbios não fariam diferença já que não poderiam ser relacionados a outras *features*. Contudo, se escolhermos representar nossos documentos através de N-gramas, esses termos farão diferença, já que uma *feature* se torna automaticamente a junção de ‘n’ palavras, assim, maximizando as chances de o N-grama manter o contexto original.

A etapa de retirada de stopwords tem o potencial de reduzir o número de *features* em um documento, já que elimina palavras irrelevantes. Por consequência, melhora a precisão de nosso classificador além de seu desempenho em função do tempo. Na imagem 20a retiramos os advérbios da lista de stopwords, na 20b retiramos as stopwords de nossos documentos e na imagem 20c mostramos como identificamos os advérbios por meio de suas *tags.*

```
.....
a)  stopWordsWithoutAdverbs = []
    for stopWord in stopWords:
        if getTagWord(stopWord) not in LIST_OF_RELEVANT_TAGS:
            stopWordsWithoutAdverbs.append(stopWord);
.....
b)  for comment in comments:
        commentFiltered = [];
        for word in comment:
            if word not in stopWordsWithoutAdverbs:
                commentFiltered.append(word);
        commentsFiltered.append(commentFiltered);
```

```
......
c)  LIST_OF_RELEVANT_TAGS = ['RB','RBR','RBS'];
```

Imagem 20 - a) Retirada dos advérbios da lista de stopwords, b) Retirada das stopwords dos documentos de texto, c) Definindo tipos de advérbios

4.3.2.2 Stemming em features

Nessa etapa, reduzimos as nossas *features* para uma forma comum. Em geral, esse tratamento retira sufixos e prefixos mantendo apenas a raiz (*steam*) da palavra. Como já comentando nesse documento, o objetivo é representar palavras em sua forma canônica.

O potencial resultado dessa etapa de processamento depende da quantidade de variações que um mesmo termo pode ter em diferentes textos. A principal importância dessa etapa está ligada ao fato de que palavras de mesmo radical puderam agora ser facilmente encontradas em diferentes documentos, melhorando o desempenho dos nossos classificadores. A imagem 21 mostra quais são as funções utilizadas para realizar o stemming de nossas *features* em nosso algoritmo de tratamento.

```
from nltk.stem import *
from nltk.stem.snowball import SnowballStemmer
.....
for comment in comments:
    commentStemmizing = []
    for word in comment:
        commentStemmizing.append(stemmer.stem(word));
    commentsStemmizing.append(commentStemmizing);
```

Imagem 21 - Aplicação de stemming nas palavras

4.3.2.3 Features em formato N-grama

Nessa etapa de pré-processamento, cada comentário agora é representado como um N-grama de ordem “N” definido através de variáveis de comando. Caso não queiramos usar essa técnica, a função de conversão em N-grama é chamada para N = 1. Assim, como resultado, teremos tuplas de 1 palavra cada. Tuplas de uma (1) palavra têm o mesmo potencial de exibição das próprias palavras. Apenas mudamos a estrutura com que essas features são representadas. Essa transformação é necessária para que o nosso algoritmo de pré-processamento possa trabalhar com uma estrutura padronizada. Para n > 1 a função será chamada gerando tuplas de ordens no intervalo [n > 0 e n < n-escolhido]. A imagem 22 mostra como esse processo foi realizado por nosso algoritmo.

```
from nltk import ngrams
.....
for comment in comments:
        newCommentList = [];
        for n in range(1,ordem+1):
                nGrams = ngrams(comment, n)
                for gram in nGrams:
                        newCommentList.append(gram);
        newCommentsList.append(newCommentList);
```

Imagem 22 - Representando textos com N-gramas

4.4 Indução de Classificadores usando WEKA

Cada arquivo de saída do algoritmo de pré-processamento representa uma configuração diferente de técnicas utilizadas e foi usado como entrada para a ferramenta WEKA. Dessa forma, obtemos resultados diferentes para analisar (Capitulo 5). Através de sua interface, escolhemos qual arquivo usar como entrada, qual algoritmo usar para classificação e como dividiremos nossa base de dados entre treinamento testes.

Cada base de documentos será dividida pelo WEKA entre dados de treinamento e testes usando holdout. Nessa abordagem, uma porcentagem especifica dos documentos já classificados serão utilizados para treinar nossos classificadores e a outra parte para testes. O capítulo 5 a seguir traz detalhes dos resultados obtidos neste trabalho.

5. Experimentos e resultados

Nessa seção, definimos nossa abordagem para escolha dos experimentos, como representamos cada experimento e como será o processo de cada um. Na seção 5.1 falamos um pouco sobre como cada caso de teste pode variar, ou seja, suas principais diferenças em relação aos outros. Na seção 5.2 apresentamos os classificadores que serão experimentados e na seção 5.3 os resultados de cada teste realizado, comparando os resultados para as técnicas empregadas para cada classificador.

5.1. Instâncias de arquivos de treinamento e testes

Para testes, usaremos as variações de saída explicitadas na imagem 23. Dessa forma avaliaremos o impacto que cada técnica de tratamento obteve na classificação dos documentos. Cada variação de fluxo do algoritmo gerará arquivos de saída arff, que serão usados como entrada na ferramenta WEKA.

Os arquivos de saída do algoritmo de pré-processamento são gerados e o fluxo especifico de código que eles representam foi representado em seus nomes. Dessa forma, o nome do arquivo será composto por 3 caracteres, cada um trazendo uma informação das técnicas que foram ou não utilizadas no pré-processamento (veja imagem 23). Ao final, escolhemos 16 casos de teste (imagem 24).

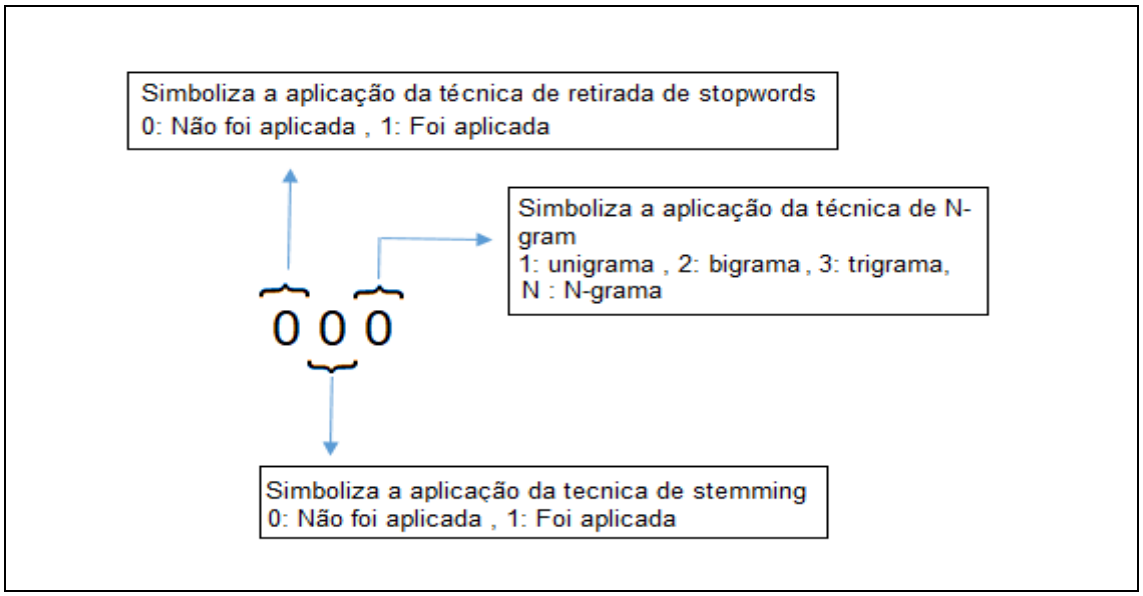

Imagem 23 - Representação do nome da Instância de arquivo de teste

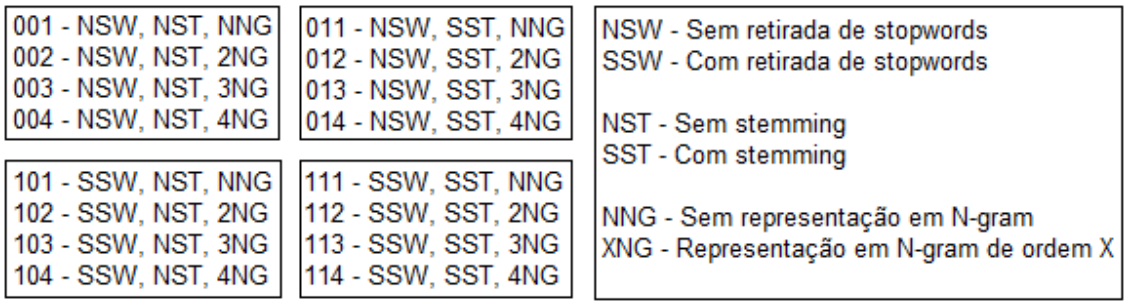

Imagem 24 - Instâncias de Teste

5.2. Realizando experimentos com WEKA

Cada combinação de arquivo arff e classificador será testada separadamente, ou seja, para cada caso de teste (combinação) a ferramenta WEKA irá fazer um processamento à parte. O trabalho de selecionar arquivos e configurações para cada caso de teste foi feito manualmente. Os classificadores escolhidos para nossos testes foram: Naive Bayes, Suport Vector Machine (SVM) e Random Forest. Na Imagem 25 os classificadores escolhidos na ferramente WEKA.

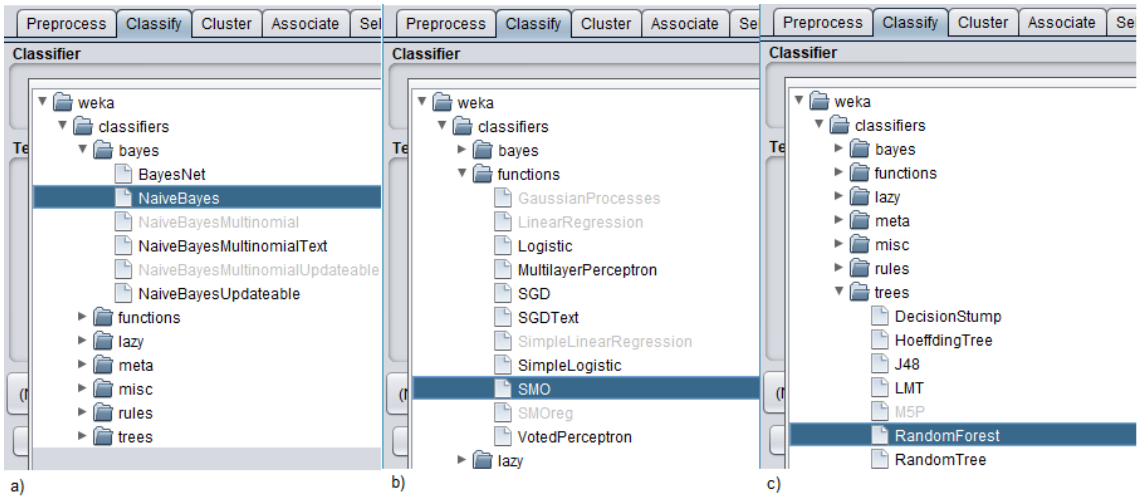

Imagem 25 - Escolha: a) Naive Bayes, b) SVM, c) Random Forest

Fonte: Criadas a partir do uso da ferramenta WEKA

Além disso, também foi usada a técnica de holdout. Essa funcionalidade nos permite escolher como dividir nossa base entre treinamento e testes, por exemplo, para 300 documentos e um holdout de 50%, usaremos 150 documentos para treinar o algoritmo e 150 para testes. Para nossos

experimentos adotamos uma divisão de 70-30, ou seja, 70% para treinamento e 30% para testes.

5.3. Resultados

Avaliaremos o desempenho de nossos classificadores a partir das instancias de teste usando as seguintes métricas: Acurácia, F-Measure e AUC. Essas foram as métricas escolhidas, já que representam bem a combinação de todas as outras já citadas nesse documento.

5.3.1. Resultados do classificador Naive Bayes

Instância	Acurácia	F-Measure	AUC
001	0,833	0,828	0,933
002	0,833	0,828	0,942
003	0,778	0,765	0,950
004	0,700	0,662	0,955
101	0,878	0,876	0,939
102	0,844	0,840	0,948
103	0,811	0,801	0,948
104	0,767	0,749	0,944
011	0,856	0,854	0,957
012	0,856	0,852	0,961
013	0,778	0,765	0,958
014	0,744	0,721	0,958
111	**0,900**	**0,899**	0,957
112	0,867	0,865	0,967
113	0,833	0,826	**0,966**
114	0,800	0,789	0,956

Tabela 3 - Resultados para o classificador Naive Bayes

Podemos observar na tabela 3 que os melhores resultados foram obtidos quando usamos em paralelos as técnicas de retirada de stopwords e stemming (**resultados em negrito**). Embora o ganho tenha sido modesto, esses resultados mostram que nossas técnicas de pré-processamento tiveram um papel importante na etapa de classificação.

Outro fator interessante em nossos resultados é que, à medida que aumentamos o “N” do N-grama, perdemos acurácia e F-Measure, mas, na

maioria das vezes, ganhamos em AUC. Isso significa que as features no formato N-grama levaram a uma classificação errada de algumas instancias, mas, na média, ajudaram na classificação, pois essa configuração aumentou a capacidade do classificador em distinguir as classes.

É possível notar também que as técnicas de stemming e retirada de stopwords se mostraram vantajosas quando usadas separadamente uma da outra, melhorando os resultados de maneira parecida. Os menores valores para AUC, acurácia e f-measure foram obtidos sem o uso das técnicas da retirada de stopwords e stemming (resultados sublinhados), mais uma vez demonstrando a relevância dessas técnicas para nosso processo de classificação com o classificador Naive Bayes. Os valores altos de AUC significam que o naive Bayes tem forte capacidade de ranqueamento e de classificar corretamente cada classe.

5.3.2. Resultados do classificador Suport Vector Machine

Instância	Acurácia	F-Measure	AUC
001	0,889	0,889	0,888
002	0,889	0,889	0,888
003	0,889	0,889	0,888
004	0,889	0,889	0,888
101	**0,911**	**0,911**	**0,909**
102	0,900	0,900	0,897
103	0,889	0,889	0,887
104	0,889	0,889	0,887
011	0,900	0,900	0,899
012	0,889	0,889	0,887
013	0,867	0,866	0,865
014	0,867	0,866	0,865
111	0,900	0,900	0,899
112	0,889	0,889	0,887
113	0,878	0,878	0,876
114	0,889	0,889	0,887

Tabela 4 - Resultados para classificador Suport Vector Machine

A tabela 4 apresenta os resultados da classificação das instâncias de testes para o classificador SVM. Como podemos observar, as técnicas de N-grama e stemming não ajudaram nos resultados, a técnica N-grama piorou os resultados em praticamente todas as instâncias, a técnica de stemming também não significou melhor proveito do classificador. Quando os dois tratamentos foram aplicados ao mesmo tempo, o classificador apresentou os piores resultados (resultados sublinhados).

Já a técnica de retirada de stopwords ajudou o classificador a apresentar os melhores resultados (**resultados em negrito**). No geral, podemos observar que o algoritmo SVM mantém sua estabilidade (baseado na variação das métricas de avaliação) e resultados promissores independente das técnicas aplicadas, mostrando apenas uma pequena melhora para a técnica de retirada de stopwords.

5.3.3. Resultados do classificador Random Forest

Instância	Acurácia	F-Measure	AUC
001	0,844	0,844	0,939
002	0,889	0,889	0,958
003	0,844	0,842	0,960
004	0,878	0,877	0,946
101	**0,933**	**0,933**	**0,974**
102	0,922	0,922	0,957
103	0,867	0,867	0,955
104	0,878	0,878	0,935
011	0,878	0,878	0,963
012	0,889	0,889	0,972
013	0,878	0,877	0,957
014	0,856	0,855	0,954
111	0,922	0,922	0,969
112	0,900	0,900	0,970
113	**0,933**	**0,933**	**0,979**
114	0,889	0,888	0,959

Tabela 5 - Resultados para o classificador Random Forest

Para o algoritmo Random Forest, os resultados obtidos assim como os resultados do SVM mostraram pequenas variações. O maior impacto foi obtido a partir da retirada das stopwords (**resultados em negrito**), uma pequena melhora também foi obtida a partir da aplicação da técnica de stemming (**resultados em negrito**). Já a técnica de N-grama resultou em bastante variação nos resultados, às vezes melhorando, e outras vezes piorando o desempenho do classificador.

De toda forma, os piores resultados foram encontrados para Instâncias de testes onde as técnicas de pré-processamento não foram aplicadas (resultados sublinhados), o que mostra a importância desses tratamentos para os resultados do classificador Random Forest.

5.3.4. Comparando classificadores

Na tabela 6 apresentamos os melhores resultados para cada classificador e quais técnicas de pré-processamento foram usadas para a obtenção desses resultados. Da mesma forma, na Tabela 7 apresentamos os piores resultados.

Classificador	Acurácia	F-Measure	AUC	Instância
Naive Bayes	0,900	0,899	0,957	101
SVM	0,911	0,911	0,909	101
Random Forest	**0,933**	**0,933**	**0,974/0,979**	**101/113**

Tabela 6 - Comparando os melhores resultados

O algoritmo Random Forest apresentou os resultados mais promissores (**resultados em negrito**), chegando até 93,3% de acurácia e 97,9% de AUC, além de 93,3 % de F-measure. Em segundo lugar, ficou o classificador Suport Vector Machine com 91,1% de acurácia, 91,1% de F-Measure e 90,9% de AUC. Em terceiro lugar o classificador Naive Bayes com 90% de acurácia, 89,9% de F-Measure e 95,7% de AUC. Podemos notar que os melhores resultados foram obtidos a partir da aplicação da técnica de retirada de stopwords sem a aplicação de stemming e sem N-grama, com excessão do algoritmo Random Forest que conseguiu manter os excelentes resultados mesmo para o uso das três técnicas.

Classificador	Acurácia	F-Measure	AUC	Instância
Naive Bayes	0,700	0,662	0,955	004
SVM	0,867	0,866	0,865	013/014
Random Forest	0,844	0,844	0,939	001

Tabela 7 - Comparando os piores resultados

Entre os piores resultados, Naive Bayes se destaca negativamente (resultados sublinhados) por ficar bem abaixo dos concorrentes (70% de acurácia e 66,2% de F-Measure), Suport Vector Machine e Random Forest mostraram resultados muito próximos, O classificador SVM apresenta maior Acuracia e F-Measure (86,7% e 86,8%), mas, o Random Forest apresentou maior AUC.

A nível de técnicas de pré-processamento, observamos que os piores resultados aparecem quando não usamos as técnicas de retirada de stopwords e quando usamos técnicas de N-grama para um “n” grande (maior que 3). Já para a técnica de stemming, os resultados variam de acordo com o classificador avaliado.

5.3.5. Considerações finais

Entre todos os fatores observados, destacamos aqui os mais importantes. A retirada de stopwords mostrou ser vantajosa em todos os casos e apresentou os melhores resultados para cada classificador. Os algoritmos SVM e Random Forest impressionam por manter uma determinada constância em seus resultados, entre eles, o SVM mostrou variações mínimas e ótimos resultados deixando bem claro que sua capacidade de classificar corretamente Instâncias dependem menos de técnicas de pré-processamento do que seus concorrentes.

Já o classificador Random Forest mesmo apresentando os melhores resultados, apresentou uma variação maior para as técnicas de tratamento utilizadas. Consideramos os resultados dos classificadores SVM e Random Forest como promissores, o primeiro por manter bons resultados em diferentes cenários e o segundo por apresentar os melhores resultados mesmo que dependendo mais das técnicas de pré-processamento.

Já para o classificador Naive Bayes, as técnicas se mostraram mais relevantes, já que, à medida que eram usadas, faziam com que o classificador apresentasse melhores resultados. Além disso, mostrou os piores resultados de acurácia e F-measure, resultados esses, bem abaixo da média quando comparado com os outros classificadores. Por outro lado, mostrou grande capacidade de ranqueamento. Para sistemas que tem como objetivo ordenar os documentos classificados pode valer a pena usar esse classificador aplicando técnicas de pré-processamento para melhorar as outras medidas. Em geral concluímos:

● **Naive Bayes** depende das técnicas de pré-processamento para apresentar melhores resultados de acurácia e f-measure. Mas, apresenta bons valores para a medida AUC.

● **SVM** se mostrou constante em seus bons resultados para todas as métricas de avaliação independente das técnicas de pré-processamento usadas para tratar o texto. Entretanto, os valores menores de AUC mostram que esse classificador não é tão bom ordenador como o Random Forest e o Naive Bayes.

● **Random Forest** apresenta os melhores resultados em todas as métricas de avaliação usadas e varia de forma pouco significante (mas, maior do que o SVM) para aplicação de técnicas de pré-processamento.

● A **retirada de stopwords** foi a técnica mais impactante no desempenho dos classificadores e esteve presente nos melhores resultados apresentados.

● O **stemming** apresenta impacto razoável nos classificadores Naive Bayes e SVM, mas, pouco significante para o Random Forest.

● A **técnica de N-grama** apresenta pequenas melhoras quando o “N” é pequeno (N = 2) e pioras quando o valor de “N” cresce muito.

6. Conclusão

Nesse trabalho foi apresentado o uso de técnicas de pré-processamento de texto aplicado em *reviews* de smartphones. Os comentários tratados foram usados para extração de sentimentos com o intuito de classificar corretamente a polaridade de cada comentário. Os testes foram realizados usando a ferramenta WEKA e testamos três (3) classificadores diferentes, comparando seus resultados a partir de algumas métricas de avaliação.

6.1. Contribuições e Dificuldades

O trabalho em questão oferece estatísticas relacionadas ao impacto da aplicação de técnicas de pré-processamento em documentos para classificação de sentimentos. Pudemos comparar diferentes técnicas de pré-processamento, buscando entender como elas se comportam quando são aplicadas sozinhas ou em conjunto umas com as outras. Também pudemos verificar a importância dessas técnicas para diferentes classificadores. Esses resultados nos ajudaram a entender como cada classificador se comporta e o quão relevante é o pré-processamento de texto aplicado a classificadores de aprendizagem supervisionada.

Os comentários foram colhidos e classificados de forma manual e esse processo se mostrou desgastante, além disso foi comum achar erros de grafia, excesso de pontuação, entre outros problemas já conhecidos para tratamento de linguagem natural. Esses tipos de erros, considerados ruídos, invalidaram a escolha de alguns comentários para nosso corpus, deixando claro a necessidade de uma etapa de tratamento de ruído para o processo de coleta automático de comentários em futuros trabalhos.

6.2. Trabalhos futuros

Em trabalhos futuros pretendemos automatizar o processo de obtenção de comentários por meio de Api(s), além de automatizar o processo de classificação prévia através de informações como “número de estrelas”. A partir disso testaremos se esse processo automatizado ofereceu uma boa base de dados para nossos algoritmos classificadores.

Também é pretendido aplicar o processo descrito nesse projeto para outros contextos em ambientes diferentes (redes sociais por exemplo) e verificar até que ponto esse processo se mostra eficiente na classificação de comentários.

Referências Bibliográficas

[1] Liu, B.; Hu, M.; and Cheng, J.; Opinion observer: Analyzing and comparing opinions on the web. In WWW '05: Proceedings of the 14th international conference on World Wide Web, pages 342–351, New York, NY, USA. 2005. ACM.

[2] T. Nasukawa and J. Yi, "Sentiment analysis: Capturing favorability using natural language processing," Proceedings of the Conference on Knowledge Capture (K-CAP), 2003.

[3] Liu, B.; Sentiment Analysis and Subjectivity. In Handbook of Natural Language Processing, Second Edition, N. Indurkhya and F.J. Damerau, Editors. 2010.

[4] Lucas Vinicius Avanço. Sobre normalização e classificação de polaridade de textos opinativos na web. USP – São Carlos Outubro de 2015

[5] Liu, B. (2012). Sentiment analysis and opinion mining. In Synthesis Lectures on Human Language Technologies, volume 5, páginas 1–167. Morgan & Claypool Publishers.

[6] Pang, B. e Lee, L. (2008). Opinion mining and sentiment analysis. In Foundations and trends in information retrieval, volume 2, páginas 1–135. Now Publishers Inc

[7] Wilson, T., Wiebe, J., e Hwa, R. (2004). Just how mad are you? finding strong and weak opinion clauses. In AAAI 2004, volume 4, páginas 761–769.

[8] Hu, M. e Liu, B. (2004). Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, páginas 168–177. ACM

[9] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha. Comparing and combining sentiment analysis methods. In Proceedings of the 1st ACM Conference on Online Social Networks (COSN'13), 2013.

[10] [Taboada et al., 2011] Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). Lexicon-based methods for sentiment analysis. Comput. Linguist., 37(2):267– 307.

[11] Benevenuto F., Ribeiro F. e Araújo M. (2017) Métodos para Análise de Sentimentos em Mídias Sociais. http://homepages.dcc.ufmg.br/~fabricio/webmedia.

[12] Pak, A., & Paroubek, P. (2010). Twitter Sentiment Analysis and Opinion Mining. In Proceedings of the 7th Conference on International Language Resources and Evaluation, LREC '10, Valletta, Malta.

[13] Kouloumpis, E., Wilson, T., & Moore, J. (2011). Twitter Sentiment Analysis: The Good the Bad and the OMG!. In Proceedings of the 5th International AAAI Conference on Weblogs and Social Media.

[14] Hanhoon Kang, Seong Joon Yoo, Dongil HanSenti-lexicon and Improved Naïve Bayes Algorithms for Sentiment Analysis of Restaurant Reviews Expert Syst Appl, 39 (2012), pp. 6000-6010

[15] G. Vinodhini and R. Chandrasekaran, "Sentiment Analysis and Opinion Mining: A survey," International Journal, vol. 2, no. 6, 2012.

[16] Feldman, 2013 R. Feldman. Techniques and Applications for Sentiment Analysis Communications of the ACM, 56 (4) (2013), pp. 82-89

[17] Crabtree, C, Golder, M, Gschwend, T, Indridason, IH (n.d.). Campaign sentiment in European Party Manifestos. Unpublished manuscript.

[18] Fábio Ricardo Araújo da Silva, Detecção de Ironia e Sarcasmo em Língua Portuguesa: Uma Abordagem Utilizando Deep Learning. 2018. Monografia Universidade Federal de Mato Grosso, Instituto de Computação, Cuiabá, 2018

[19] Dickerson, J. P., Kagan, V., & Subrahmanian, V. S. (2014, August). Using Sentiment to Detect Bots on Twitter: Are Humans More Opinionated Than Bots?. In Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on (pp. 620-627). IEEE.

[20] Brito EMN. Mineração de Textos: Detecção Automática de Dentimentos em Comentários nas Mídias Sociais[dissertação]. Belo Horizonte: Fundação Mineira de Educação e Cultura; 2017.

[21] SCHÜTZE, Hinrich; MANNING, Christopher D.; RAGHAVAN, Prabhakar. Introduction to Information Retrieval. Cambridge University Press, 2008.

[22] ARMANO, Giuliano; FANNI, Francesca; GIULIANI, Alessandro. Stopwords Identification by Means of Characteristic and Discriminant Analysis. In: Proceedings of the International Conference on Agents and Artificial Intelligence-Volume 2. SCITEPRESS-Science and Technology Publications, Lda, 2015. p. 353-360...

[23] NPM, Npm stopword, Disponível em: https://www.npmjs.com/package/stopword. Acesso em dezembro de 2018

[24] PYTHON, Python stemming, Disponível em: https://pythonprogramming.net/stemming-nltk-tutorial/. Acesso em dezembro de 2018

[25] MOURA, Maria Fernanda et al. Um modelo para a seleção de n-gramas significativos e não redundantes em tarefas de mineração de textos. Embrapa Informática Agropecuária-Boletim de Pesquisa e Desenvolvimento (INFOTECA-E), 2010

[26] RISH, Irina et al. An empirical study of the naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence. New York: IBM, 2001. p. 41-46.

[27] GAMALLO, Pablo; GARCIA, Marcos. Citius: A naive-bayes strategy for sentiment analysis on english tweets. In: Proceedings of the 8th international Workshop on Semantic Evaluation (SemEval 2014). 2014. p. 171-175.

[28] KANG, Hanhoon; YOO, Seong Joon; HAN, Dongil. Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews. Expert Systems with Applications, v. 39, n. 5, p. 6000-6010, 2012.

[29] ORHAN, Emin. Cover's Function Counting Theorem (1965). 2014.

[30] MULLEN, Tony; COLLIER, Nigel. Sentiment analysis using support vector machines with diverse information sources. In: Proceedings of the 2004 conference on empirical methods in natural language processing. 2004.

[31] BREIMAN, L. Random forests. Machine Learning, v. 45, n. 1, p. 5–32, Out 2001. ISSN 1573-0565.

[32] BASTOS, Denise GD; NASCIMENTO, Patricia S.; LAURETTO, Marcelo S. Proposta e análise de desempenho de dois métodos de seleção de Caracterıśticas para Random Forests. IX Simpósio Brasileiro de Sistemas de Informação, p. 49-60, 2013.

[33] WEKA, Data Mining Software, Disponível em : https://www.cs.waikato.ac.nz/ml/weka/. Acesso em dezembro de 2018

[34] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

[35] PYSCIENCE, O que é ? Por que Usar ? Disponível em: http://pyscience-brasil.wikidot.com/python:python-oq-e-pq. Acesso em dezembro de 2018

[36] AMAZON, Disponível em: https://www.amazon.com/. Acessado em dezembro de 2018

[37] SAATY, Thomas L.; VARGAS, Luis G. Diagnosis with dependent symptoms: Bayes theorem and the analytic hierarchy process. Operations Research, v. 46, n. 4, p. 491-502, 1998.