

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

ADRIANO FRANCISCO KULTZAK

**CATEGORIZAÇÃO DE TEXTOS UTILIZANDO ALGORITMOS DE
APRENDIZAGEM DE MÁQUINA COM WEKA**

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2016

ADRIANO FRANCISCO KULTZAK

**CATEGORIZAÇÃO DE TEXTOS UTILIZANDO ALGORITMOS DE
APRENDIZAGEM DE MÁQUINA COM WEKA**

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Marcos Vinicius Fidelis

PONTA GROSSA

2016



Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
COORDENAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS



TERMO DE APROVAÇÃO

CATEGORIZAÇÃO DE TEXTOS UTILIAZANDO ALGORITMOS DE APRENDIZAGEM DE MÁQUINA COM WEKA

por

ADRIANO FRANCISCO KULTZAK

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 3 de junho de 2016 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Marcos Vinicius Fidelis
Prof. Orientador

Helyane Bronoski Borges
Membro titular

Geraldo Ranthum
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Dedico este trabalho à minha família,
pelos momentos de ausência.

AGRADECIMENTOS

Expresso minha gratidão a todas aquelas pessoas envolvidas neste trabalho. Peço desculpas àquelas que não estão presentes entre essas palavras, mas que fazem parte do meu pensamento e de minha gratidão.

Agradeço ao meu orientador Prof. Marcos Vinicius Fidelis, pela sabedoria e colaboração com que me guiou nesta trajetória.

Aos demais professores do curso e meus colegas de sala.

A Coordenação do curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, com quem pude contar com o apoio durante esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

A linguagem é um processo de criação livre; suas leis e princípios são fixos, mas a maneira com a qual os princípios de geração são usados é livre e infinitamente variada. Mesmo a interpretação e uso de palavras envolve um processo de criação livre.
(CHOMSKY, Noam, 1999)

RESUMO

KULTZAK, Adriano Francisco. **Classificação de textos utilizando algoritmos de aprendizagem de máquina com WEKA**. 2016. 74 f. Trabalho de Conclusão de Curso - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

Este trabalho apresenta um estudo sobre a classificação de conjuntos de dados textuais utilizando algoritmos de aprendizagem de máquina através da plataforma WEKA. Apresenta-se uma visão geral sobre a interpretação da linguagem nos textos, as características e etapas da classificação e formas de avaliação dos resultados e desempenho dos algoritmos. Através de testes exemplifica-se todo o processo de classificação de textos e pela análise dos resultados procuram-se possíveis afinidades entre diversos algoritmos através da aplicação da técnica em bases de dados com características diversas.

Palavras-chave: Classificação. Aprendizagem de máquina. WEKA. Conjuntos de dados.

ABSTRACT

KULTZAK, Adriano Francisco. **Classificação de textos utilizando algoritmos de aprendizagem de máquina com WEKA**. 2016. 74 f. Trabalho de Conclusão de Curso - Federal Technology University - Paraná. Ponta Grossa, 2016.

This paper presents a study on the classification of textual data sets using machine learning algorithms through the WEKA platform. It presents an overview of the interpretation of language in texts, characteristics and stages of the classification rating and forms of results evaluation and performance of the algorithms. Through testing it exemplifies up the process of classification of texts and analyzing the results looking for affinities between various possible algorithms by the technique of application databases with different characteristics.

Keywords: Classification. Machine Learning. WEKA. Data sets.

LISTA DE ILUSTRAÇÕES

Figura 1 - Tela Inicial do WEKA.....	36
Figura 2 - Interface Explorer do WEKA.....	37
Figura 3 - Interface Experimenter do WEKA.....	38
Figura 4 - Interface Knowledge Flow do WEKA.....	38
Figura 5 - Interface SimpleCli do WEKA.....	39
Figura 6 - Tela de configuração do filtro StringToWordVector.....	40
Figura 7 - Filtro de seleção de atributos.....	41
Figura 8 - Estrutura de diretórios que serão convertidos.....	44
Figura 9 - Comando para conversão de diretório em .arff.....	45
Figura 10 - Arquivo .arff obtido após a conversão dos diretórios.....	48
Figura 11 - Visualização da ocorrência dos termos em cada instância.....	49
Figura 12 - Tela de resultado da classificação da base de dados 20 Newsgroups....	51
Figura 13 - Matriz de confusão gerada pelo algoritmo de aprendizagem.....	52
Figura 14 - Predições do algoritmo por instância.....	52
Figura 15 - Diretórios com arquivos de texto.....	69
Figura 16 - Arquivos de texto.....	69
Figura 17 - Predições do algoritmo por instância.....	70
Figura 18 - Arquivo .arff gerado.....	70
Figura 19 - Abertura do arquivo .arff através doWEKA Explorer.....	71
Figura 20 - Tela StringToWordVector.....	72
Figura 21 - Último atributo corresponde às classes.....	72
Figura 22 - Opções para classificação.....	73
Figura 23 - Resultado da classificação.....	74
Figura 24 - Predições do algoritmo selecionado.....	74
Gráfico 1 - Composição dos conjuntos de dados utilizados.....	44
Gráfico 2 - Composição dos conjuntos de dados pré IG/GR/OR.....	51
Gráfico 3 - Composição dos conjuntos de dados pós IG/GR/OR.....	51
Gráfico 4 - Número de 1ª colocações nas bases de dados DB_World.....	63
Gráfico 5 - Número de 1ª colocações nas bases de dados OH.....	63
Quadro 1 - Classificações possíveis para um documento.....	35

LISTA DE TABELAS

Tabela 1 - Características das bases de dados utilizadas.....	43
Tabela 2 - Conjuntos particionados com Percentage Split 66% dados aleatórios.....	53
Tabela 3 - Conjuntos particionados com Percentage Split 66% dados aleatórios.....	53
Tabela 4 - Conjuntos particionados com Percentage Split 66% ordem preservada.....	54
Tabela 5 - Conjuntos particionados com 10 fold Cross-validation.....	54
Tabela 6 - Resultados InfoGain com Percentage Split 66% dados aleatórios.....	55
Tabela 7 - Resultados 20 Newsgroups dados aleatórios.....	55
Tabela 8 - Resultados 20 Newsgroups atributos igualados.....	55
Tabela 9 - Resultados InfoGain com Percentage Split 66% dados aleatórios.....	56
Tabela 10 - Resultados InfoGain com Percentage Split 66% ordem preservada.....	56
Tabela 11 - Resultados InfoGain com 10 fold Cross-validation.....	57
Tabela 12 - Resultados GainRatio com dados aleatórios.....	57
Tabela 13 - Resultados GainRatio com ordem preservada.....	57
Tabela 14 - Resultados GainRatio com Cross-validation.....	58
Tabela 15 - Resultados InfoGain SubsetEval com dados aleatórios.....	58
Tabela 16 - Resultados InfoGain SubsetEval com ordem preservada.....	59
Tabela 17 - Resultados InfoGain SubsetEval com Cross-validation.....	59
Tabela 18 - Resultados OneR com dados aleatórios.....	60
Tabela 19 - Resultados OneR com ordem preservada.....	60
Tabela 20 - Resultados OneR com Cross-validation.....	60
Tabela 21 - Número de vezes em cada posição por algoritmo.....	61

LISTA DE SIGLAS

ARFF	Attribute-Relation File Format
EM	Expectation Maximization
ENIAC	Electronic Numeric Integrator And Computer
FN	False Negative
FP	False Positive
GB	Giga byte
GHz	Giga Hertz
HD	Hard Drive
IBK	Instance-based nearest neighbor
KDD	Knowledge Discovery and Data Mining
KNN	K-nearest neighbors
NLP	Natural Language Processing
QP	Quadratic Programming
RAM	Random Access Memory
SMO	Sequential Minimal Optimization
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
WEKA	Waikato Environment for Knowledge Analysis

SUMÁRIO

1 INTRODUÇÃO.....	13
1.1 OBJETIVOS.....	13
1.1.1 Objetivo Geral.....	14
1.1.2 Objetivos Específicos.....	14
1.2 JUSTIFICATIVA.....	14
2 FUNDAMENTAÇÃO TEÓRICA.....	16
2.1 LINGUAGEM NATURAL.....	16
2.1.1 Processamento da Linguagem Natural.....	17
2.2 APRENDIZAGEM DE MÁQUINA.....	18
2.2.1 Aplicações.....	29
2.2.2 Aprendizagem de Máquina na Mineração de Dados.....	21
2.3 A CATEGORIZAÇÃO DE TEXTOS.....	21
2.3.1 Etapas para categorização.....	23
2.3.2 Classificação Supervisionada.....	29
2.3.3 Classificação Supervisionada X Não Supervisionada.....	29
2.3.4 Algoritmos para classificação de textos.....	30
2.3.5 Avaliação dos classificadores.....	34
2.4 WEKA.....	36
2.4.1 Interfaces.....	36
2.4.2 Recursos para Classificação.....	39
3 METODOLOGIA.....	43
3.1 COLETA DE DADOS.....	43
3.2 PREPARAÇÃO E PREPROCESSAMENTO.....	45
3.3 SELEÇÃO DE ALGORITMOS.....	46
3.4 TREINAMENTO E CLASSIFICAÇÃO.....	47
3.4.1 <i>Cross-validation</i>	47
3.4.2 <i>Percentage Split</i>	47
4 RESULTADOS.....	48
4.1 RESULTADOS DA COLETA DE DADOS.....	48
4.2 RESULTADOS DO PREPROCESSAMENTO.....	59
4.3 RESULTADOS DA CLASSIFICAÇÃO.....	51
5 CONCLUSÃO.....	63
5.1 TRABALHOS FUTUROS.....	64
REFERÊNCIAS.....	65
APÊNDICE A – PROCESSO DE CLASSIFICAÇÃO DE TEXTOS COM WEKA SIMPLIFICADO.....	69

1 INTRODUÇÃO

A capacidade humana de se comunicar pode ter sido fundamental para que a espécie homo sapiens prevalecesse em relação às outras que coexistiram em nossa origem, (HARARI, 2014). O ser humano atual através de sua inteligência vem evoluindo esta capacidade no decorrer dos seus 150 mil anos de existência.

As máquinas, mais especificamente os computadores modernos ao considerarmos como marco a criação do ENIAC em 1947 (MARTIN, 1995), mesmo sendo desenvolvidas a menos de 100 anos estão cada vez mais parecidas conosco inclusive nos recursos da inteligência. Nossa interação e utilização com melhor proveito dos recursos tecnológicos depende inevitavelmente da capacidade das máquinas, que possuem o sistema binário como linguagem, de interpretar a nossa linguagem “natural” seja ela falada ou escrita.

Quase todo o nosso conhecimento é disponibilizado em publicações diversas como livros, revistas, jornais, músicas, filmes e já fizemos tanta coisa que para um indivíduo ter acesso a um tipo de conhecimento específico seria muito difícil caso não tivéssemos inventado também a internet, recurso que viabiliza acesso a grande parte dos arquivos produzidos por nós e local onde diariamente é produzida uma quantidade imensa de dados.

A partir da grande disponibilidade de dados no formato de texto, da capacidade atual de uma máquina aprender podendo contar com ferramentas específicas já desenvolvidas e principalmente em prol do avanço nas áreas de descoberta do conhecimento e interpretação da linguagem natural os estudos deste trabalho foram desenvolvidos.

No decorrer deste trabalho será apresentado o processo de classificação de textos através da utilização dos algoritmos de aprendizagem de máquina presentes no ambiente de aprendizagem WEKA, contando com exemplificação teórica e resultados práticos do procedimento.

1.1 OBJETIVOS

Os objetivos deste trabalho estão definidos de forma geral e específica nas seções seguintes.

1.1.1 Objetivo Geral

Demonstrar os resultados da aplicação da coleção de algoritmos de aprendizagem de máquina e ferramentas para pré-processamento, classificação e visualização do ambiente para análise de conhecimento, na categorização de conjuntos de dados textuais, bem como apresentar as partes que compõem o processo de categorização através dessas ferramentas, e analisar de forma comparativa as características como precisão e acurácia que resultarem da aplicação dos algoritmos de classificação utilizados.

1.1.2 Objetivos Específicos

- Descrever as etapas do processo de categorização de bases de dados textuais, exemplificando através do registro dos resultados obtidos pela aplicação prática de algoritmos diversos e recursos da ferramenta WEKA (*Waikato Environment for Knowledge Analysis*) em bases de dados com características variadas.
- Descrever algumas formas de avaliação utilizadas para comparação do desempenho de classificadores.
- Obter informações relativas ao desempenho de diferentes tipos de classificadores de texto em bases de dados textuais.
- Apresentar e identificar possíveis padrões no comportamento de conjuntos de dados com características diversas durante a classificação como possíveis afinidades entre tipo de conjunto de dados e tipo de algoritmo.
- Demonstrar técnicas utilizadas atualmente para melhorar os resultados obtidos nas bases de dados classificadas, identificando as diferenças de resultados anteriormente e posteriormente a sua aplicação.

1.2 JUSTIFICATIVA

Para auxiliar na atual pesquisa, desenvolvimento e otimização do estudo da classificação de texto, este trabalho apresenta uma visão geral do processo e demonstra através da utilização de diversas bases de dados e diversos algoritmos

os resultados que podem ser obtidos com a utilização das técnicas atuais, servindo como elemento de observação para quem pesquisa a área da descoberta de conhecimento e linguagem natural através da utilização dos recursos da ferramenta WEKA (HALL et al., 2009), logo, resultados da aplicação de diversos classificadores relacionados a diversas bases de dados podem auxiliar outras pesquisas principalmente no momento da escolha da ferramenta adequada ou no melhoramento de ferramentas existentes.

O diagnóstico é uma etapa importante para a otimização de um processo, tendo em vista que a percepção das possibilidades facilita uma melhor exploração, e a identificação de comportamentos pode viabilizar o aprimoramento na utilização de bases de dados textuais contribuindo para o processo final de transformação de informação em conhecimento (SETZER, 1999)..

Informação que constitui “uma abstração informal (isto é, não pode ser formalizada através de uma teoria lógica ou matemática), que representa algo significativo para alguém através de textos, imagens, sons ou animação” (SETZER, 1999) e conhecimento “uma abstração interior, pessoal, de alguma coisa que foi experimentada por alguém” (SETZER, 1999). Pode-se relacionar a grande quantidade de informação disponível na forma de textos a um potencial conhecimento aplicável seja por humanos com os avanços no estudo da linguagem natural, seja pelas máquinas com o desenvolvimento da aprendizagem.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho está centrada nos estudos da descoberta de conhecimento em bases de dados textuais utilizando algoritmos de aprendizagem de máquina com abrangência do campo da Linguagem Natural e conceitos teóricos do funcionamento da ferramenta WEKA.

Nas seções seguintes serão abordados os conceitos de Linguagem Natural, Aprendizagem de Máquina e suas aplicações, bem como a definição de Categorização de Textos através de suas etapas e formas de avaliação, por fim será apresentada a ferramenta de classificação WEKA.

2.1 LINGUAGEM NATURAL

A linguagem natural é um campo estudado em diversas áreas como computação, psicologia e linguística. A relação entre a quantidade de conhecimento adquirido e a quantidade de informação a que a pessoa está exposta é tratada no “problema de Platão” o qual invoca o questionamento de como uma criança pode formar sentenças com menor ou maior complexidade diante de estímulos pobres.

Hauser, Chomsky e Fitch (2002) tratam a linguagem como um mecanismo inato da espécie humana, onde não aprendemos a partir da inexistência de conhecimento prévio. Quando estamos aumentando nossa capacidade de nos comunicar, na verdade, estamos apenas ajustando parâmetro que já estão estabelecidos em nossas funções cognitivas. Apesar da discutibilidade dessa proposta ela nos auxilia a compreender como uma criança independente da língua, religião ou cultura mesmo com pouca informação se torna plenamente capacitada a se comunicar.

Quando uma criança está aprendendo a falar ela não segue regras gramaticais como as vistas em uma sala de aula através da relação aluno-professor, no momento em que assimila o título de seu desenho favorito, essa criança também não entrou em contato com nenhum tipo de ensinamento sobre as regras da comunicação escrita, através de exemplos como esses temos noção do quanto a linguagem está associada a processos os quais já nascemos sabendo e por mais

que estes exemplos não constituam um entendimento absoluto, eles nos ajudam a compreender o funcionamento da assimilação da linguagem natural em humanos.

Ao estudar a aprendizagem de máquina e o processamento da linguagem natural, depara-se com o problema de como transcrever as regras que regem a comunicação da forma como um ser humano interpreta para a forma como uma máquina o faz. De acordo com Chowdhury (2003), para explicar como computadores podem ser utilizados para manipular a linguagem natural, surgiu a área de pesquisa do Processamento da Linguagem Natural ou NLP (*Natural Language Processing*). Os pesquisadores dessa área estudam a forma como os humanos entendem e utilizam a linguagem para que dessa forma ferramentas e técnicas sejam desenvolvidas para fazer sistemas computacionais entenderem e manipularem linguagens naturais desenvolvendo as tarefas desejadas.

2.1.1 Processamento da Linguagem Natural

A construção de programas que entendem a linguagem natural envolve três problemas principais: o primeiro está relacionado com o pensamento, o segundo com a representação e significado das entradas linguísticas, e o terceiro está relacionado com o conhecimento mundial. Dessa forma um sistema NLP pode começar no nível das palavras – através da determinação da origem e estrutura – então pode passar para o nível das sentenças – para determinar a ordem das palavras, gramática e significado das sentenças – então para o contexto do ambiente geral. Uma palavra pode ter um significado em determinado contexto e pode estar relacionada a diversas outras (CHOWDHURY, 2003).

De acordo com Feldman (1999) para o entendimento da Linguagem Natural é necessário distinção entre os sete níveis utilizados por pessoas para extrair significado de textos ou linguagem escrita:

- Nível Fonético ou Fonológico: o qual trata da pronúncia;
- Morfológico: que trata das menores partes da palavra que carregam significado;
- Sintático: que trata da estrutura gramatical das sentenças;
- Semântico: que trata do significado das palavras e sentenças;
- Discursivo: que trata da estrutura dos diferentes tipos de texto;

- Pragmático: que lida com o conhecimento advindo do mundo fora do conteúdo dos documentos;

Chowdhury (2003) divide os trabalhos e pesquisas relacionados ao processamento da linguagem natural em três categorias:

- Análise léxica e morfológica;
- Semântica e análise do discurso;
- Abordagens baseadas no conhecimento e ferramentas para NLP.

2.2 APRENDIZAGEM DE MÁQUINA

A palavra aprender em sua origem latina é composta por “*ad*” que significa junto e por “*prae*” (à frente) + “*hendere*” relacionado a hera, uma planta trepadeira que se prende as paredes para poder crescer, seu sentido original era o de levar para junto de si metaforicamente levar para junto da memória (HARPER, 2016).

O processo da aprendizagem nos seres vivos é resultado de interações entre estruturas mentais e o meio ambiente, ela se torna fundamental para a adaptação ao meio em que estão expostos. No ramo da psicologia diversas abordagens são tomadas por estudiosos para representar os conceitos de aprendizagem dos seres humanos, segundo Skinner (1950), “aprendizagem é uma mudança na probabilidade de uma resposta específica”, este conceito pode ser aplicado para seres vivos e também para máquinas a partir do momento em que um dispositivo artificial seja capaz de retornar respostas a entradas de dados, da mesma forma que seres vivos respondem a estímulos do meio.

Uma calculadora, por exemplo é um dispositivo que retorna uma resposta a partir da inserção de dados, porém não gostaríamos que a calculadora aprendesse a realizar as contas de uma maneira diferente retornando resultados diferentes, mas sim que ela aprendesse de maneira autônoma novos cálculos que possam solucionar velhos problemas apurando a probabilidade da solução.

Como em outras disciplinas a aprendizagem de máquina possui aspectos teóricos e empíricos, sustentando-se no fato de que a maioria dos algoritmos de aprendizagem são muito complexos para uma análise formal, tem-se um componente empírico fortalecido durante o estudo dos algoritmos. Experimentos envolvem sistematicamente variar uma ou mais variáveis independentes e examinar

seus efeitos nas variáveis dependentes, assim realizam-se diversas iterações da etapa de aprendizagem para que sejam medidas os aspectos do comportamento do algoritmo sob diferentes condições (LANGLEY, 1988).

Segundo Langley (1988) muitas definições de aprendizado baseiam-se na noção de aumento de performance, assim várias medidas de performance são variáveis dependentes naturais para os experimentos de aprendizagem de máquina, assim como para o estudo do aprendizado em humanos. As medidas de performance são o princípio para a avaliação de qualquer método de aprendizagem de máquina pois outras medidas como por exemplo a de entendibilidade do método podem ser utilizadas de forma informativa porém não relevante, observando-se que em alguns casos métodos intuitivamente plausíveis de aprendizagem resultam em piores performances.

2.2.1 Aplicações

Entre as razões encontradas para a pesquisa da aplicação de aprendizagem de máquina em problemas do mundo real podemos também citar que aplicações conceituais possuem baixa aceitação na hora de demonstrar resultados e que aplicações do mundo real além de motivar a pesquisa contínua se beneficiam da sinergia entre pesquisa e prática motivando ambos (SEGRE, 1992).

Jason Brownlee (2013) escreveu um artigo com 10 exemplos de aplicações que mostram como a aprendizagem de máquina é utilizada atualmente:

- Detecção de Spam: somos todos familiares com esse exemplo presente nas caixas de e-mail onde os spams são automaticamente identificados e movidos para a caixa apropriada;
- Identificação de fraude em cartões de crédito: baseado nas transações de um consumidor dentro do mês, são identificadas quais foram realizadas por ele e quais não foram e automaticamente gerar um estorno das transações não comprovadamente geradas pelo dono do cartão.
- Reconhecimento de dígitos: identificar os dígitos escritos em códigos postais dos envelopes para a distribuição por região;
- Identificação de voz: Utilizado em assistentes virtuais como os do celular identificar pedidos específicos feitos através da voz por um usuário;

- Detecção facial: Identificar imagens que apresentam rostos para organizar por pessoa, ou no caso de redes sociais possibilitar marcação pelos usuários, alguns modelos de câmeras digitais também possuem essa capacidade.

- Recomendação de produtos: analisar o histórico de compras de um usuário para identificar quais produtos ele poderia se interessar em comprar. Além de diversos sites de venda online redes sociais também possuem um sistema de recomendação de usuários.

- Diagnóstico médico: de acordo com os sintomas de um paciente cruzados com uma base de dados detectar se um paciente possui determinada doença, auxiliando o diagnóstico do médico.

- Mercado de ações: identificar oportunidades de compra e venda de ações baseadas nos movimentos atuais e anteriores no preço delas, servindo como suporte para a decisão dos analistas;

- Segmentação de consumidores: analisar o comportamento de um consumidor durante o período de testes de um produto para identificar aqueles que irão adquirir a versão paga ou não auxiliando em decisões como solicitar uma conversão para a versão paga ou ativar intervenções para persuadir o consumidor.

- Identificação de formas: identificar formas conhecidas em desenhos feitos a mão para criação de diagramas ou para identificação de escrita, presente em alguns aplicativos para celulares.

Entre os problemas apresentados acima estão alguns dos mais complexos do campo da Inteligência Artificial como o processamento da linguagem natural ou a visão computacional (BROWNLEE, 2013).

Quando é escolhido o algoritmo que será usado em nossa aplicação deve-se levar em conta três variáveis principais, representação, avaliação e otimização. A primeira variável trata de como o algoritmo pode ser representado em uma linguagem computacional e quais são as entradas possíveis para o algoritmo. A avaliação é necessária para que possamos distinguir a eficiência de cada algoritmo através de medição. Finalmente, a otimização trata de se adaptar e conseguir o melhor resultado do classificador possível (DOMINGOS, 2012).

2.2.2 Aprendizagem de Máquina na Mineração de Dados

A mineração de dados ou descoberta de conhecimento em bancos de dados conhecida como KDD (*Knowledge Discovery in Databases*) que trata da descoberta de informação útil presente em grandes bases de dados através da identificação de regras e padrões nesses conjuntos que na maioria das vezes possuem especificidades, sendo assim, as ferramentas de mineração devem ser utilizadas de forma interativa e não automática gerando diversas etapas para o processo como as listadas (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996):

- Identificar o Domínio
- Preparação dos dados
- Mineração
- Processamento dos resultados
- Utilização dos resultados

A aprendizagem de máquina durante a mineração de dados acaba sendo combinada com as áreas de estatística e bancos de dados, apesar de haver controvérsias e considerações de similaridade entre as duas pode-se citar entre diferença da aprendizagem de máquina e KDD que na primeira área há uma busca pelas estruturas que estão por trás dos dados e contribuem para sua formação enquanto no KDD os dados são a base para o estudo e o interesse nos resultados não depende de estruturas por trás desses dados e apesar da aprendizagem de máquina ser parte do núcleo da mineração de dados elas são coisas diferentes. (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

2.3 A CATEGORIZAÇÃO DE TEXTOS

Textos são uma forma natural para armazenar informação, sua mineração possui potencial comercial porém é uma tarefa difícil e multidisciplinar, as disciplinas estudadas nesse trabalho serão as relacionadas especificamente com a classificação dos textos.

A classificação de textos gera interesse tanto em pesquisadores da área de aprendizagem de máquina e recuperação de informação quanto em desenvolvedores de aplicações que precisam trabalhar com grandes quantidades de

documentos, fator potencializado pela conectividade aumentada e disponibilidade de diversas bases de dados contendo documentos de diferentes tipos e com variadas quantidades de informação (SEBASTIANI, 2005).

A recuperação de informação e pesquisa em bases de dados com grande quantidade de textos pode ter duas aproximações, na primeira constroem-se ferramentas robustas para a pesquisa nessas bases ou na segunda solução constroem-se ferramentas robustas para estruturar esses dados tornando a pesquisa mais simples (SEBASTIANI, 2005).

Apesar de ser encontrada uma barreira para a aprendizagem de máquina ao considerar o processamento da linguagem natural, uma vez que as sentenças que formamos não são aleatórias. Utiliza-se como um meio eficiente a representação das palavras através do espaço vetorial (MANNING; RAGHAVAN e SCHÜTZE, 2009).

Existem duas variações da classificação de textos, na clusterização os documentos são agrupados dentro de estruturas latentes porém ainda não detectadas dentro do repositório e na segunda forma a classificação dentro de categorias é feita através de uma entrada, em outras palavras tem-se na primeira forma uma classificação sem conhecimento das entradas desejadas e na segunda as classes são predefinidas e servem de entrada. (SEBASTIANI, 2005).

Clusterização tem sido usada para melhorar os conjuntos de treinamento através da descoberta de “estruturas” nesses conjuntos expandindo-os com novos atributos extraídos dos *clusters*, é usada também como alternativa a seleção de atributos e para propagar informação que possui classe definida através da clusterização de documentos rotulados e não rotulados em conjunto. (KYRIAKOPOULOU, KALAMBOUKIS, 2006). De acordo com Hotho, Nürnberger, e Paaß (2005), a clusterização pode ser utilizada para a formação de grupos de documentos com conteúdo similar, os chamados *clusters*, com conteúdo mais similar dentro do *cluster* e menos similar entre os diferentes *clusters*.

Ainda segundo Sebastiani (2005), inicialmente em meados da década de 70 as aplicações construídas objetivavam a indexação automática de textos para sistemas de recuperação de informações booleanos, tendo em vista os termos de indexação serem predefinidos podemos considerar uma forma de classificação onde utilizamos analogamente tais expressões como classes dos sistemas atuais, com o crescimento de importância e quantidade de documentos nas décadas de 80 e 90

surgiram aplicações de classificação de texto como os filtros de notícias, classificação de patentes e classificação de páginas WEB.

Um exemplo prático atual da tecnologia tema deste trabalho está na “Detecção de relacionamentos entre categorias utilizando classificação de textos” (MENGLER; GOHARIAN, 2010). O artigo que trata da descoberta de relacionamentos entre conceitos úteis que não são aproveitados quando os conceitos estão separados, como no exemplo do câncer de pulmão associado a fumantes, que não tem proveito quando esses dois temas aparecem separados, para isso com a ajuda da classificação de textos identificam esses relacionamentos entre diferentes categorias e os representam em uma estrutura de dados.

2.3.1 Etapas para categorização

A classificação de documentos textuais pode ser enquadrada em duas etapas distintas, a primeira fase é a recuperação de informações a partir dos textos a serem analisados, nela, dados numéricos são extraídos da base textual. A segunda fase trata da classificação em si, fase onde ocorre o processamento realizado para que o algoritmo decida em qual categoria o texto deve pertencer. Fases adicionais são incluídas para que o esforço computacional seja reduzido ou para que o algoritmo passe por um treinamento com uma amostra de dados antes da classificação desejada. (MAHINOVSKÝ et al., 2007).

Segundo Carrilho (2007) na metodologia da mineração de textos o processo mais oneroso está no pré-processamento visto que não há uma técnica que obtenha uma representação satisfatória em todos os domínios, sendo assim a representação adequada é alcançada de forma empírica.

2.3.1.1 Tokenização

Ao analisar-se a quantidade de elementos que um texto contém pode-se identificar diversos componentes recorrentes sejam eles palavras, pontuação, ou símbolos, durante o processo de tokenização (em inglês, *tokenizing*) estes elementos são separados e listados, apenas palavras são interessantes para os processos posteriores logo pontuação, espaços ou delimitadores de tabulação

servem como separadores durante a formação dos *tokens*, que são instâncias compostas por uma sequência de caracteres em um documento em particular que é agrupada na forma de uma unidade semântica útil para o processamento. (MANNING; RAGHAVAN e SCHÜTZE, 2009).

Segundo CARRILHO (2007) apesar de a identificação de *tokens* ser uma tarefa simples para o ser humano, ela pode se tornar complexa quando executada por um computador pelo fato de os delimitadores assumirem diferentes papéis. Por exemplo o “ponto”, que é utilizado para marcar o fim de uma sentença ou na composição de uma abreviatura.

Uma sentença pode ter seus vocábulos dispostos em diferentes ordens as quais podem até mesmo modificar seu significado, são consideradas colocações as palavras que apresentam essa relação. Por exemplo “provocar essa destruição” e “essa destruição foi provocada”, por isso é interessante que a tokenização seja composta por todo o conjunto de palavras que podem traduzir uma ideia diferente (SOARES, 2008).

2.3.1.2 Remoção de *stopwords*

Algumas palavras não oferecem relevância durante o processo de classificação e devem ser removidas na etapa de pré-processamento, palavras como conectores, preposições, artigos, pronomes e palavras comuns de grande ocorrência e com pouco valor para a classificação, constituem uma lista dos *tokens* com pouca relevância que variam de acordo com a língua do texto e serve como parâmetro para a formação do dicionário de palavras definitivas para as demais etapas.

A remoção das *stopwords* pode liberar espaço nos vetores e em alguns casos melhorar a performance do classificador ajudando na recuperação de informação, a maioria dos termos utilizados são relativamente óbvios e pertencem a um conjunto não muito amplo de palavras, contudo é importante ressaltar que algumas palavras que aparecem nos documentos tem maior valor sintático do que o valor semântico que interessa para o classificador. Pode-se considerar uma *stopword* aquela encontrada quando consulta-se um documento mas não é

relevante se a consulta está ou não relacionada com o documento (WILBUR, SIROTKIN, 1992)

2.3.1.3 *Stemming*

Palavras que possuem variações pequenas como tempo verbal ou mudanças entre plural e singular passam por uma transformação para que seja reduzida a ocorrência de *tokens* com sentidos similares aumentando a eficiência durante a atribuição dos pesos para os termos (LOVINS, 1968).

O trabalho do algoritmo responsável pelo *stemming* é remover as variações de uma palavra reduzindo para uma mesma raiz ou uma forma comum, pesquisadores na área de recuperação de informações podem considerar esse passo importante por diversas razões. Na análise morfológica automática, a raiz da palavra pode ter menor interesse imediato que seus sufixos, assim como a contagem da frequência dos termos pode ser relevante para análise matemática ou estatística de um corpus, o que requer expressões idênticas, mas alguns problemas linguísticos são comuns para qualquer algoritmo que realize *stemming* não importando sua finalidade (LOVINS, 1968).

De acordo com o algoritmo utilizado para *stemming* o resultado nem sempre será um termo com validade gramatical, pois podem ser cortados letras do final ou reduzidos a pedaços da palavra original sem sentido para leitura (WEINBERG, 2016).

2.3.1.4 Seleção de Atributos

Manter o foco nas informações mais relevantes mesmo diante de uma quantidade de dados muito grande se tornou uma importante tarefa relacionada a aprendizagem de máquina. Há grande quantidade de atributos em exemplos encontrados geralmente no material científico ou corporativo que é mais comum no contexto da aprendizagem de máquina, com a ajuda da Web o processo acabou sendo inundado com material de baixa qualidade de informação se torna importante o foco na seleção dos atributos mais relevantes para a representação dos dados (BLUM; LANGLEY, 1997).

A literatura da aprendizagem de máquina apresenta diferentes conceitos de relevância por que esta geralmente depende do objetivo estudado, na sequência 5 conceitos de relevância que são importantes na tarefa de seleção de atributos (BLUM; LANGLEY, 1997).

- Relevante ao alvo: ao alterar o valor do atributo para determinado documento alvo então a classe a qual estava atribuído também é alterada.

- Fortemente relevante à amostra: o atributo deve necessariamente estar presente na amostra que contém as classes A e B. É semelhante ao relevante ao alvo mas relacionado a amostra.

- Fracamente relevante à amostra: é considerado fracamente relevante a amostra caso possa ser removido um subconjunto de atributos e ele se torne fortemente relevante.

- Relevante como uma medida de complexidade: compões o menor número de atributos necessários para atingir a ótima performance.

- Útil de forma incremental: caso produza uma melhora na acurácia quando inserido no subconjunto de atributos.

Em tarefas de classificação de textos geralmente utiliza-se métodos de filtragem que não dependem do algoritmo que será utilizado para a classificação, nesse processo as características do próprio conjunto de treinamento são utilizadas para remover atributos irrelevantes e selecionar atributos importantes (BLUM; LANGLEY, 1997).

A forma mais simples de indexação dos atributos selecionados em textos é tratar cada palavra como um atributo, porém palavras possuem características como sinônimos ou significados múltiplos que complicam sua indexação, o que motivou o surgimento de métodos mais complexos para seleção de atributos em textos, como a indexação de frases, onde atributos são extraídos a partir da presença de uma ou mais palavras em um trecho sintático do texto ou o método de clusterização de termos, onde grupos de atributos podem ser substituídos por um único atributo correspondente a sua soma lógica ou numérica (LEWIS,1997).

Técnicas como a apresentada por Nigam et al. (2000) tentam reduzir a necessidade de dados pré classificados, que muitas vezes são escassos, durante a fase de seleção de atributos através de Maximização de expectativa ou EM (*Expectation-Maximization*) que é a utilização de documentos não classificados ou

incompletos, visto que os documentos não classificados carecem de classe, em conjunto com documentos rotulados para aumentar a posterior eficiência do classificador. Para que os documentos não rotulados não acabem prejudicando a etapa de treinamento é proposto no trabalho citado a inclusão de um fator peso nesses documentos, outra proposição para aumentar a eficiência da técnica é modelar as classes de acordo com uma mistura múltipla de componentes.

O uso da seleção de atributos não é exclusivo dos problemas de classificação, mas particularmente nas tarefas de classificação faz sentido a utilização de classes para supervisionar essa etapa para assegurar que os atributos melhor relacionados as classes sejam selecionados (AGGARWAL, e ZHAI, 2012). Alguns métodos de seleção de atributos utilizados em tarefas de classificação são apresentados abaixo:

2.3.1.4.1 Gini-index

É a probabilidade condicional de um documento pertencer a uma classe por conter determinada palavra, é um dos métodos mais comuns para discriminar o peso de um termo de acordo com a equação (1) (AGGARWAL; ZHAI, 2012).

$$G(w) = \sum_{i=1}^k p_i(w)^2 \quad (1)$$

Os valores do Gini-index estarão sempre entre $1/k$ e 1 sendo k o número de classes diferentes para o termo w , quanto maior o valor de $G(w)$ maior é o poder discriminativo da palavra w em relação a determinada classe. Uma ressalva feita com relação a essa abordagem é que suas medidas podem não refletir um valor discriminativo preciso dos atributos pois a distribuição global da classe pode variar e interferir no peso, porém pode ser construído um Gini-index normalizado (AGGARWAL; ZHAI, 2012), como visto em (2).

$$p'_i(w) = \frac{p_i(w)/P_i}{\sum_{j=1}^k p_j(w)/P_j} \quad (2)$$

2.3.1.4.2 Information Gain

Outra forma de identificar o poder discriminativo de um termo é a através do *Information Gain* (Ganho de informação) $I(w)$ descrito na fórmula (3), utilizando a noção de entropia que caracteriza a impureza de determinado conjunto de exemplos e a medida de informação que representa o oposto ou a pureza de um conjunto de classes.

$$I(w) = - \sum_{i=1}^k P_i \cdot \log(P_i) + F(w) \cdot \sum_{i=1}^k P_i p_i(w) \cdot \log(p_i(w)) + (1 - F(w)) \cdot \sum_{i=1}^k P_i p_i(w) \cdot \log(1 - p_i(w)) \quad (3)$$

Dado $P(i)$ como a probabilidade global da classe i e $P_i(w)$ a probabilidade da classe i em um documento que contém a palavra w e $F(w)$ a fração de documentos que contém o termo. Quanto maior o valor do *Information Gain* em determinado termo, maior seu poder de discriminação (AGGARWAL; ZHAI, 2012).

2.3.1.4.3 Mutual information

É uma maneira formal de modelar a quantidade de informação entre os atributos e as classes. A quantidade de informação mútua entre o termo e a classe serve como base para identificar uma possível coocorrência entre a classe e o termo (AGGARWAL; ZHAI, 2012).

$$M_{avg}(w) = \sum_{i=1}^k P_i \cdot M_i(w) \quad (4)$$

$$M_{max}(w) = \max_i \{M_i(w)\}$$

A informação mútua $M_i(w)$ é específica de de uma classe em particular, logo, é necessário que se compute a informação mútua geral como uma função da

informação mútua de um termo w nas diferentes classes. Isso é feito através da utilização dos valores médio $M_{avg}(w)$ e máximo $M_{max}(w)$.

2.3.1.4.4 χ^2 - Statistic

Através da χ^2 – Statistic (5) computa-se a falta de independência entre o termo w e a classe em questão i . Assim como na medida de *Mutual Information*, ela mede a correlação entre termos e categorias, porém com a vantagem de apresentar um valor normalizado de mais fácil comparação entre palavras dentro da mesma categoria (AGGARWAL; ZHAI, 2012).

$$\chi_i^2(w) = \frac{n \cdot F(w)^2 \cdot (p_i(w) - P_i)^2}{F(w) \cdot (1 - F(w)) \cdot P_i \cdot (1 - P_i)} \quad (5)$$

Temos como n o número total de instâncias, $P_i(w)$ é a probabilidade condicional da classe i conter o termo w e $F(w)$ é a fração dos documentos que contém o termo.

2.3.2 Classificação Supervisionada

A criação de um classificador capaz de aprender a partir de um conjunto de treinamento e generalizar para as novas instâncias que aparecem é denominado aprendizagem de máquina indutivo ou supervisionado (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

2.3.3 Classificação Supervisionada X Não Supervisionada

Quando se tem acesso a conjuntos de dados já rotulados utilizados para treinamento podemos considerar que o processo já sabe o que fazer, sendo dessa forma, “supervisionado”, como um professor ensinando as respostas certas. Infelizmente encontrar esse professor é muitas vezes difícil, com custo elevado ou até mesmo impossível (DAUMÉ III, 2012).

Os casos em que permanece a necessidade de analisar os dados mesmo sem eles estarem rotulados são considerados de aprendizado não supervisionado, onde se aprende sem um professor. A chave para o aprendizado não supervisionado está em desconhecer a “resposta certa” (DAUMÉ III, 2012).

Durante a classificação supervisionada apresentam-se os possíveis resultados recolhidos através de observação, dessa forma ele possuirá um escopo limitado e pré-definido de resultados que serão utilizados como parâmetro e acima de tudo referência durante a classificação.

Ao classificar através do método não supervisionado não são apresentados exemplos da saída desejada, dessa forma o algoritmo deve possuir recursos para superar essa falta de respostas, apresentando os resultados classificados de acordo com categorias formuladas por ele próprio.

O ato de fornecer todos os vetores de entrada com as respectivas corretas classificações para um algoritmo durante o treinamento consiste no cenário ideal para a classificação, contudo, a capacidade de generalizar, ou seja, possuir condições de aplicar um classificador para entradas diferentes das quais ele viu durante o seu treinamento é entre outros fatores diretamente proporcional a sua eficiência. (DOMINGOS, 2012).

2.3.4 Algoritmos para classificação de textos

Toda informação disponível na forma de textos atualmente precisa ser organizada de alguma forma para que o acesso as partes relevantes seja facilitado ao indivíduo o qual procura, dessa forma podemos classificar esses textos de acordo com os tópicos de cada documento ou de acordo com o gênero de cada documento, sendo a primeira maneira mais direta e sumária e a segunda mais ampla por poder contemplar classificações como a forma como foram escritos os documentos ou até mesmo seu público-alvo. Durante a classificação de acordo com os gêneros, encontramos heterogeneidade nos documentos por estes serem produzidos por diversas fontes.

Na busca pela classificação de textos, pesquisadores desenvolveram diversas abordagens para se aproximar da solução ideal, entre elas podem ser citados alguns tipos de algoritmos utilizados:

2.3.4.1 Redes Neurais

As redes neurais tratam as informações aproximando-se da forma como o nosso cérebro funciona, para isso utiliza algoritmos e estruturas de dados, geralmente realiza o processamento em paralelo cada um com acesso individualizado a memória local. Ao algoritmo é ensinado uma série de regras que representam os relacionamentos dos dados, como hierarquia ou ligações possíveis, por exemplo.

A sofisticação do tratamento através das redes neurais está na tolerância a falhas, na capacidade de se auto-organizar e generalizar para que ocorra o processo de aprendizado (KRIESEL, 2007).

Uma Rede Neural composta por diversas camadas consiste em uma grande quantidade de unidades (neurônios) agrupados de acordo com um padrão de conexão, essas unidades geralmente possuem três classes, as unidades de entrada que recebem a informação, as de saída que apresentam os resultados do processamento e as unidades encontradas entre as duas anteriores conhecidas como unidades escondidas (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

Comparadas com os métodos de classificação que utilizam estatística, as redes neurais possuem tempos maiores durante o processo de treinamento do classificador.

2.3.4.2 Aprendizagem Baseada em Instâncias

Esse tipo de aprendizagem utiliza instâncias específicas em oposição aos métodos que utilizam abstrações pré compiladas durante a tarefa de classificação. Esses algoritmos também podem descrever conceitos probabilísticos pois utilizam funções de similaridade para produzir correspondências graduais entre as instâncias. (AHA; KIBLER; ALBERT, 1991).

Os algoritmos dessa classe precisam de menos tempo dedicado a fase de treinamento do que os demais tipos, porém demanda mais tempo durante o processo de classificação (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

Dentro de um espaço bidimensional o algoritmo estima qual a classificação de uma instância com base em seus vizinhos mais próximos, no caso do algoritmo

KNN. Porém pode encontrar problemas no caso de grandes bases de dados, pois o algoritmo precisa estimar qual a distância entre o elemento que deseja classificar e todos os demais elementos dos vetores de treinamento, os quais já possuem classe definida.

2.3.4.3 Algoritmos Genéticos

De forma simples, os algoritmos genéticos são procedimentos de busca probabilísticos desenvolvidos para trabalhar em grandes espaços envolvendo estados que podem ser representados por strings. Estes métodos são inerentemente paralelos e utilizam um conjunto de amostras do espaço total (população de strings) para gerar novos conjuntos de amostras. (GOLDBERG; HOLLAND, 1988).

Os sistemas de classificação que utilizam algoritmos genéticos exploram o paralelismo implícito desse método. As interações são realizadas através de mensagens. As condições são especificadas nos atributos das mensagens que eles recebem e as ações estão nos atributos das mensagens que eles enviam. O sistema resolve conflitos através de competição, não necessitando de algoritmos para o serviço, proporcionando uma capacidade incremental de inserção de novas regras sem atrapalhar as capacidades já estabelecidas. (GOLDBERG; HOLLAND, 1988).

Algoritmos genéticos têm sido utilizados para treinar o peso das Redes Neurais (SIDDIQUE; TOKHI, 2001) e para encontrar a arquitetura das redes neurais (YEN; LU, 2000).

2.3.4.4 Aprendizagem Simbólica

Baseados nos conceitos da aprendizagem simbólica da psicologia onde estuda-se a interferência da imaginação no aprendizado humano, a aprendizagem simbólica da computação faz uso de algoritmos que se baseiam nos aspectos estruturais dos dados ao contrário dos demais que se fundamentam nos aspectos numéricos (QUINLAN, 1994).

Dentre os algoritmos que utilizam aprendizagem simbólica para a construção do modelo de classificação podem ser citados as árvores de decisão como ID3 (QUINLAN, 1996).

2.3.4.5 Aprendizagem Bayesiana

De acordo com McCallum e Nigam (1998) os classificadores bayesianos também chamados de classificadores generativos tentam gerar uma classificação probabilística baseada na modelagem das características de palavras subjacentes em diferentes classes. A ideia é classificar o texto baseando-se na probabilidade deste pertencer a diferentes classes de acordo com a presença de palavras no documento. Está fundamentada na probabilidade de um elemento pertencer a determinada categoria, na aprendizagem Bayesiana o algoritmo recebe as instâncias rotuladas para o treinamento e quando uma nova instância necessita de classificação ele calcula a probabilidade desta receber cada rótulo e nomeia de acordo com a maior.

A principal vantagem do algoritmo *Naive Bayes* está na sua fase de treinamento mais curta. Além disso, seu modelo que tem a forma de um produto pode ser convertido em uma soma através do uso de logaritmos trazendo vantagens computacionais (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

2.3.4.6 Árvores de decisão

Uma árvore de decisão é construída a partir da decomposição hierárquica do espaço dos dados gerais ou de treinamento onde uma condição no valor dos atributos é utilizada para a divisão desse espaço de forma hierárquica, se tratando de documentos textuais essa condição geralmente é representada pela presença ou falta de um ou mais termos no documento. A divisão é feita de forma recursiva até que a árvore atinja uma quantidade mínima de folhas ou se chegue a uma condição estabelecida para a pureza da classe.. Em determinado documento de treinamento a sequência de classes possíveis são aplicadas na estrutura da árvore criada a partir do seu topo até que se chegue á folha mais relevante correspondente á classe. (AGGARWAL, e ZHAI, 2012).

2.3.4.7 Máquinas de Vetores de Suporte

As máquinas de vetores de suporte ou SVM (*Support vector machines*), são uma técnica de aprendizagem de máquina mais recente em comparação com a maioria, sua utilização gira em torno da noção de “margens” - os dois lados do hiperplano que sapara duas classes de dados. Através da maximização dessa margem e da separação das duas instâncias presentes em cada lado pode-se reduzir o erro de generalização (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

O custo computacional de sua fase de treinamento é elevado pois ela realiza um QP (*Quadratic Programming*) que é um tipo de problema matemático de otimização durante a criação dos Vetores de Suporte, que são compostos pelos dados de treinamento mais relevantes para o problema abordado. Utilizando uma coleção de documentos na língua portuguesa o algoritmo SVM foi comparado com Árvores de Decisão e obteve melhor desempenho para números de atributos maiores utilizados durante a etapa de aprendizagem (DA SILVA; VIEIRA, 2007).

2.3.5 Avaliação dos classificadores

Para avaliar um algoritmo de classificação devemos considerar os resultados possíveis para os documentos classificados.

TP (<i>true positive</i>)	O documento foi classificado corretamente em relação a uma categoria.
FP (<i>false positive</i>)	O documento está relacionado a uma categoria de forma incorreta.
FN (<i>false negative</i>)	O documento não está relacionado a uma categoria mas deveria estar.
TN (<i>true negative</i>)	O documento não deveria estar relacionado a determinada categoria e de fato não está.

Quadro 1 - Classificações possíveis para um documento
Fonte: Ikonomakis; Kotsiantis; Tampakas, (2005).

Os cálculos que avaliam o desempenho e servem como comparativo dos resultados dos classificadores utilizam os dados das classificações possíveis em

suas fórmulas, como por exemplo o total de documentos TP e FP na fórmula da precisão que será abordada adiante.

As avaliações baseadas em Precisão, *Recall* e Acurácia são as mais utilizadas para demonstrar os resultados da aplicação dos algoritmos e seu conceito está descrito individualmente nas seções abaixo.

2.3.5.1 Precisão

Representada por π_i , a precisão demonstra a probabilidade de um documento ser classificado dentro da categoria correta em relação a todos os documentos que foram classificados nessa categoria, sejam eles corretos ou não, de acordo com a fórmula abaixo (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005).

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \quad (6)$$

2.3.5.2 Recall

Obtém-se o *Recall* (7) como o resultado da divisão da quantidade de documentos classificados como verdadeiramente positivos pelo total de resultados que deveriam ter retornado, que abrange os verdadeiramente positivos e os falsos negativos. É a probabilidade de que se um documento aleatório tenha que ser classificado dentro de uma categoria, essa decisão será tomada. (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005).

$$\rho_i = \frac{TP_i}{TP_i + FN_i} \quad (7)$$

2.3.5.3 Acurácia

Os valores da acurácia (8) são menos relutantes a variações em relação ao número de decisões corretas do que as medidas de precisão e de *Recall* (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005).

$$A_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \quad (8)$$

2.4 WEKA

O ambiente para análise do conhecimento WEKA (*Waikato Environment for Knowledge Analysis*) foi desenvolvido pela universidade neozelandesa de *Waikato* para atender a necessidade de uma plataforma onde pode-se aplicar técnicas de aprendizagem de máquina de uma maneira unificada (HALL et al., 2009).

Na época de sua inserção em 1992 os algoritmos estavam disponíveis sempre em plataformas diferentes com formatos de dados variados e projetava-se que o WEKA não apenas iria oferecer uma caixa de ferramentas para algoritmos de aprendizagem como também uma solução para que os pesquisadores pudessem desenvolver novos algoritmos sem se preocupar com a estrutura necessária para a manipulação das bases de dados ou para a avaliação dos modelos desenvolvidos (HALL et al, 2009).

2.4.1 Interfaces

O ambiente gráfico do WEKA possui diversas interfaces para que o usuário tenha acesso as suas funcionalidades, seus recursos principais estão dispostos nas interfaces *Explorer*, *Experimenter* e *Knowledge Flow* (HALL et al., 2009).



Figura 1 - Tela Inicial do WEKA
Fonte: Autoria própria

A tela inicial conta com abas que fornecem recursos como visualização de logs de execução ou gráficos gerados pelo programa, o administrador de pacotes disponíveis para instalação, recursos de ajuda e manual.

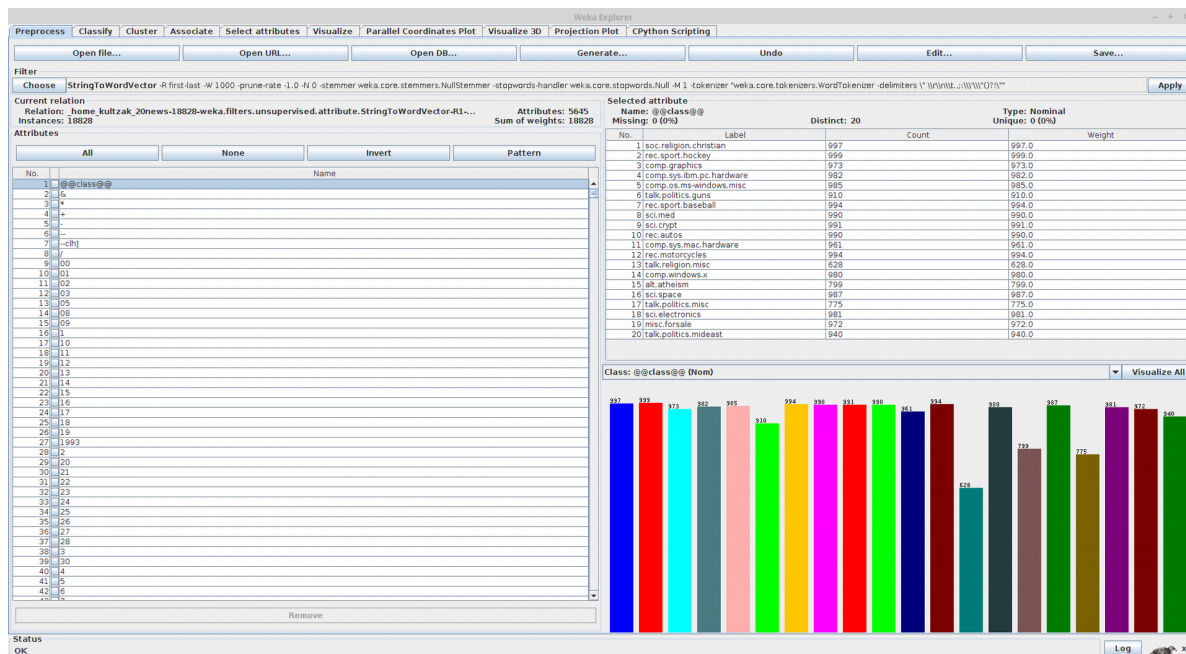


Figura 2 - Interface *Explorer* do WEKA
Fonte: Autoria própria

A interface *Explorer* do WEKA possui diversos painéis que correspondem a diferentes tarefas da mineração de dados. A primeira aba *Preprocess* é utilizada para carregar e transformar os dados através de filtros. O segundo painel *Classify* dá acesso aos algoritmos de classificação sua definição padrão é realizar *Cross-validation* no conjunto de dados preparado e além das demais opções como selecionar um conjunto de testes específico para a tarefa de classificação ainda apresenta uma representação textual do modelo utilizado para a classificação, os modelos gerados podem ser salvos para posterior visualização. As próximas abas *cluster* e *associate*, suportam respectivamente algoritmos não supervisionados e métodos de mineração por regras de associação, porém, o suporte para ambas as abas mencionadas não é tão grande quanto para algoritmos de classificação (HALL et al., 2009).

Dada a importância da seleção de atributos a próxima aba da tela *Explorer* trata da seleção de atributos, que fornece diversos algoritmos para avaliação e identificação dos atributos mais valiosos no conjunto de dados.(HALL et al., 2009). A

próxima aba *visualize* apresenta os dados através de um gráfico colorido com a opção de selecionar porções dos dados para visualizar.(HALL et al., 2009).

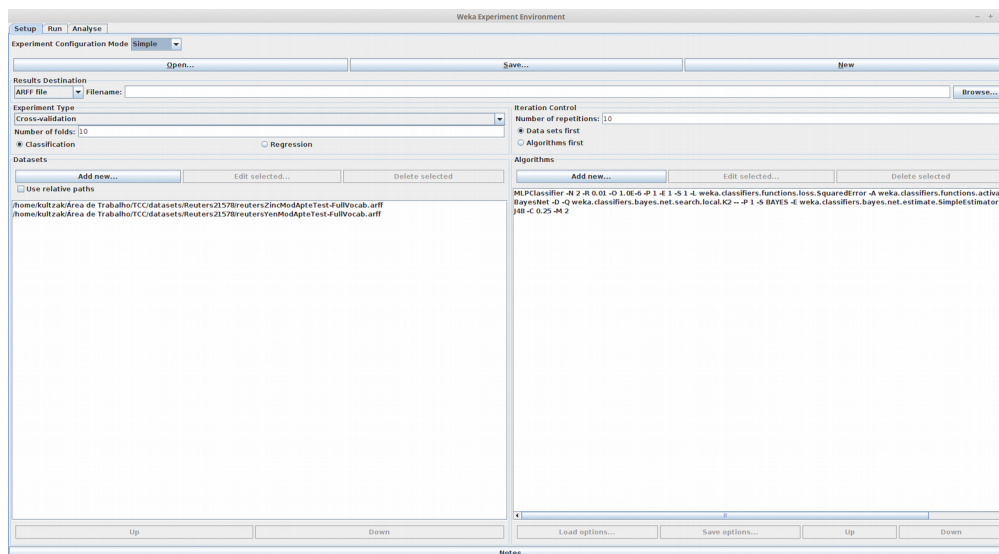


Figura 3 - Interface *Experimenter* do WEKA
Fonte: Autoria própria

A Interface *Experimenter* fornece critérios de avaliação para a comparação da performance dos algoritmos, podemos selecionar múltiplos algoritmos para serem aplicados em múltiplos conjuntos de dados. Os experimentos podem ser distribuídos entre computadores para reduzir a carga sobre uma máquina e podem ser salvos para posterior consulta, através da linha de comando pode-se executar os experimentos já criados e salvos (HALL et al., 2009).

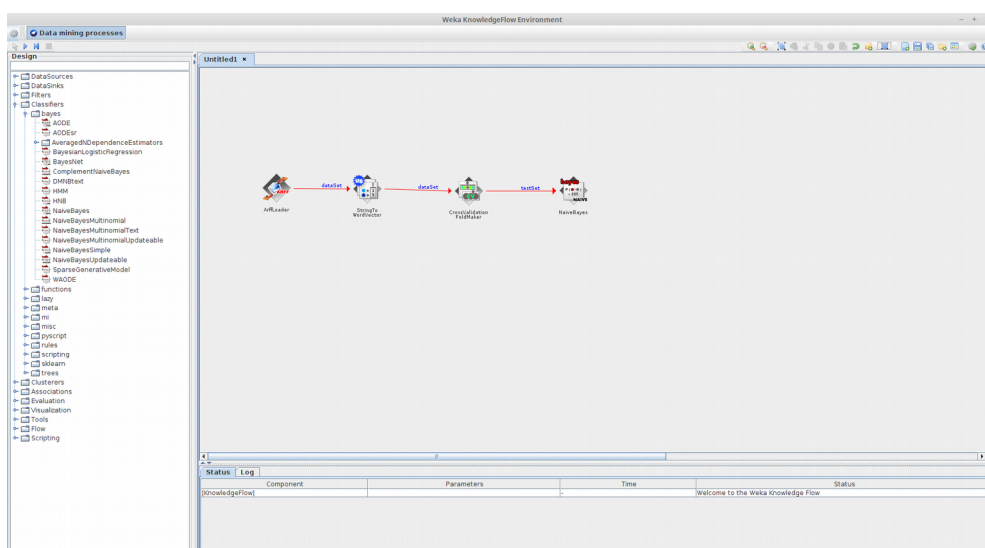
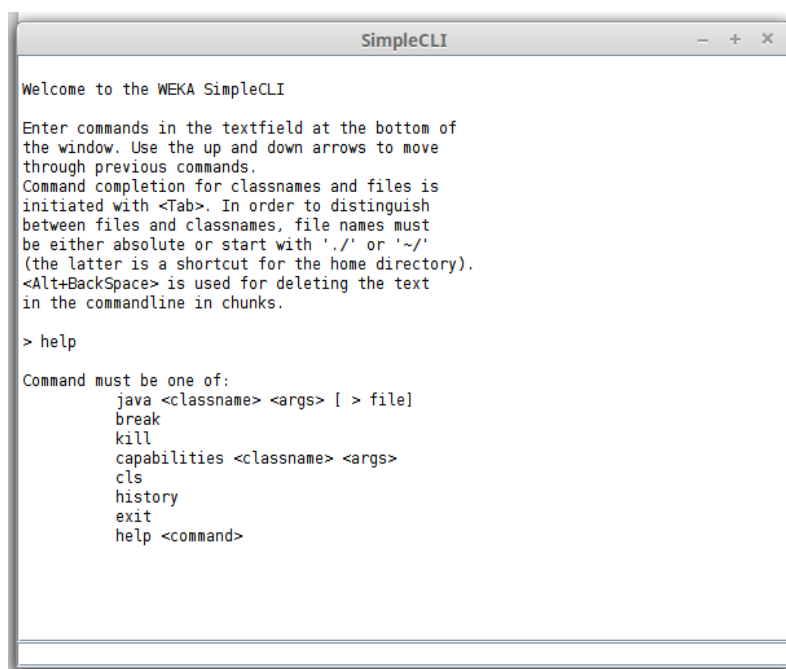


Figura 4 - Interface *Knowledge Flow* do WEKA
Fonte: Autoria própria

O *Explorer* carrega completamente os dados na memória antes do processamento, por isso se encaixa melhor com conjuntos de dados menores, como o WEKA apresenta alguns algoritmos que possuem construção incremental que podem ser acessados a partir da linha de comando ou da interface *Knowledge Flow* apresentada na Figura 3. Onde o modelo de fluxo de dados possibilita a construção de modelos incrementais que carregam dados e executam etapas de forma individual antes da aplicação dos algoritmos incrementais, o esquema com os nós interconectados pode ser salvo para reutilização ou análise (HALL et al., 2009).

2.4.2 Recursos para Classificação

Apesar das interfaces apresentadas na seção anterior implementarem os *scripts* java do WEKA de forma visual pode ser utilizada da mesma forma a interface *SimpleCli* através da linha de comando, ela inclui também alguns recursos não disponíveis em outras interfaces como o *script TextDirectoryLoader* que carrega um diretório composto por textos e transforma em um arquivo *.arff* de acordo com as especificações do comando.



```
SimpleCLI

Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.
Command completion for classnames and files is
initiated with <Tab>. In order to distinguish
between files and classnames, file names must
be either absolute or start with './' or '~/ '
(the latter is a shortcut for the home directory).
<Alt+BackSpace> is used for deleting the text
in the commandline in chunks.

> help

Command must be one of:
  java <classname> <args> [ > file]
  break
  kill
  capabilities <classname> <args>
  cls
  history
  exit
  help <command>
```

Figura 5 - Interface SimpleCli do WEKA
Fonte: Autoria própria

A estrutura do arquivo *.arff* resultante é a seguinte:

@RELATION: primeira linha do arquivo a qual contém seu nome, caso o nome possua espaços ele deve vir entre aspas.

@ATTRIBUTE: cada atributo observado no arquivo será definido nesta parte acompanhado do seu tipo, quando nominal correspondente as classes que tem seu nome **@@class@@** seguido de seus possíveis valores entre chaves.

@DATA: determina o início da seção de dados do arquivo.

Na sequência as instâncias são representadas uma em cada linha com os atributos separados por vírgulas.

Com o diretório de textos carregado e transformado em um arquivo .arff pode ser observado outro recurso útil do WEKA, o filtro *StringToWordVector* que vai quebrar a *string* obtida em um vetor com os *tokens* utilizados durante a classificação.

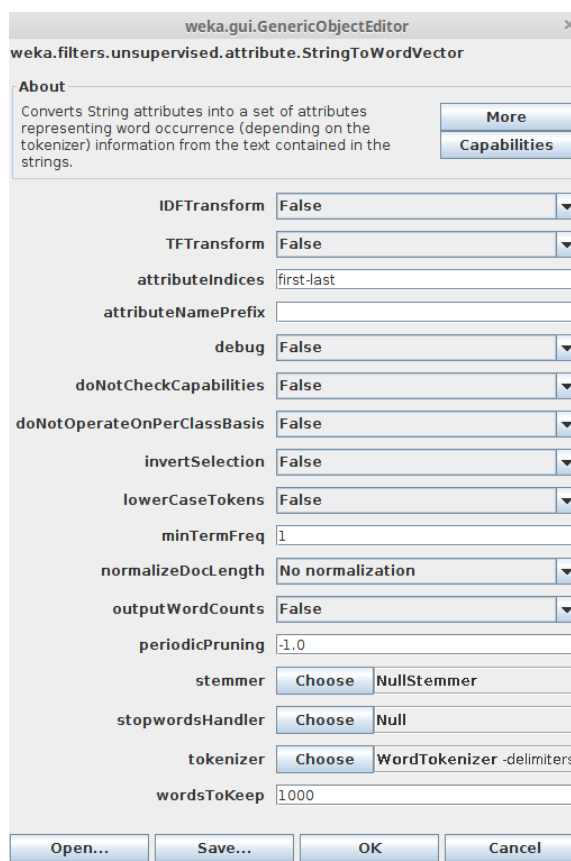


Figura 6 - Tela de configuração do filtro *StringToWordVector*
Fonte: Autoria própria

Ele conta com campos configuráveis que facilitam o trabalho com as bases de dados entre estas configurações pode-se destacar (WEINBERG, 2016):

- **outputWordCounts:** permite a inclusão da quantidade de cada termo por documento quando configurado como verdadeiro. Definido como falso por padrão,

dessa forma a informação constante no arquivo é booleana, apenas a presença ou não do termo é identificada.

- *IDFTransform* (*Inverted Document Frequency*) e *TFTransform* (*Term Frequency*): representam a frequência com que um termo aparece em um documento, quanto maior o TF e menor o IDF mais relevante ele será para determinado documento. É necessário que o *outputWordCounts* esteja configurado como *True* para que funcione.

- *normalizeDocLength*: quando ativo normaliza o peso dado a determinada palavra de acordo com o tamanho do documento que a contém, ou seja, um termo aparece a mesma quantidade de vezes em um documento extenso ou em um documento curto, logo, terá um peso maior no documento mais curto no qual é mais relevante.

- *stemmer*: permite selecionar o algoritmo de *stemming* que será aplicado durante a transformação.

- *stopwords*: seleciona uma lista de palavras não relevantes para base dados para utilizar como *stopwords*.

- *tokenizer*: seleciona o algoritmo responsável pela quebra da *string* em *tokens*.

- *minTermFreq*: determina a quantidade mínima de vezes que um termo precisa aparecer em um documento para ser aceito como *token*.

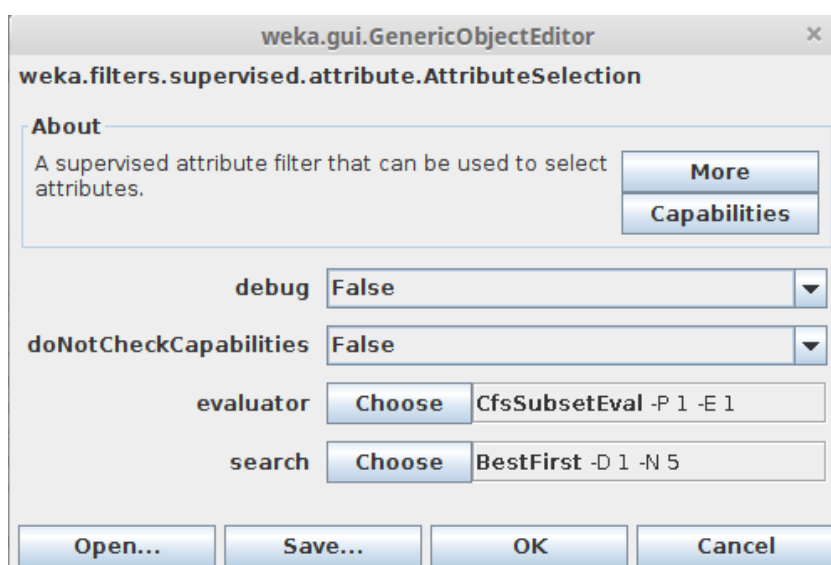


Figura 7 - Filtro de seleção de atributos
Fonte: Autoria própria

Outro recurso disponível no WEKA que pode ser utilizado durante a etapa de pré-processamento dos textos é o filtro de seleção de atributos, através dele escolhemos um avaliador e um método de buscas. O avaliador pode ser do tipo *AttributeEval* que realiza a avaliação individual do peso de cada atributo em relação aos demais ou do tipo *SubsetEval* que identifica subconjuntos de atributos que podem ser sair melhor quando selecionados em detrimento dos demais durante a etapa de classificação. Quando o avaliador é do tipo *AttributeEval* então o método de buscas selecionado deve ser o *Ranker*, que ordena os atributos de acordo com seu peso através da visualização desses valores ou de um número específico de atributos pode então ser realizada a seleção.

Um processo de classificação a partir de documentos textuais pode ser conferido no Apêndice A.

3 METODOLOGIA

A condução das atividades de classificação de textos que este trabalho trata foi desenvolvida de acordo com as etapas de, coleta de dados, preparação e pré-processamento, seleção de algoritmos, treinamento e classificação (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

Utilizou-se a versão 3.7.13 de 2015 do WEKA, instalada em um sistema operacional Linux Mint 17.2 Cinamonn 64-bit com processador I5 de 1.8 GHz, 8GB de memória RAM e 750GB de HD.

Neste capítulo será abordada durante a coleta de dados quais as características das bases selecionadas, transformações feitas durante a preparação e pré-processamento e quais as características dos algoritmos selecionados e do processo de treinamento e classificação através do WEKA.

3.1 COLETA DE DADOS

De acordo com as características deste estudo procurou-se obter bases de dados textuais que apresentassem diferenças em sua composição, quanto ao número de atributos, número de instâncias e quanto a quantidade de classes sendo binária ou composta por mais de duas classes.

A tabela abaixo apresenta as características das 9 bases de dados utilizadas.

Tabela 1 - Características das bases de dados utilizadas

Dataset	Algoritmo		
	Atributos	Instâncias	Classes
2t_movie_reviews	1166	2000	2
dbWorld_bodies	4702	64	2
dbWorld_subject	242	64	2
oh0	1003	3183	10
oh10	1050	3239	10
oh15	913	3101	10
segment_challenge	20	1500	7
smsSpam	1833	5574	2
20_newsGroups	5644	18828	20

Fonte: Autoria própria

Os conjuntos de dados apresentados foram extraídos do repositório da Universidade da Califórnia (LICHMAN, 2016) e suas características podem ser visualizadas graficamente abaixo.

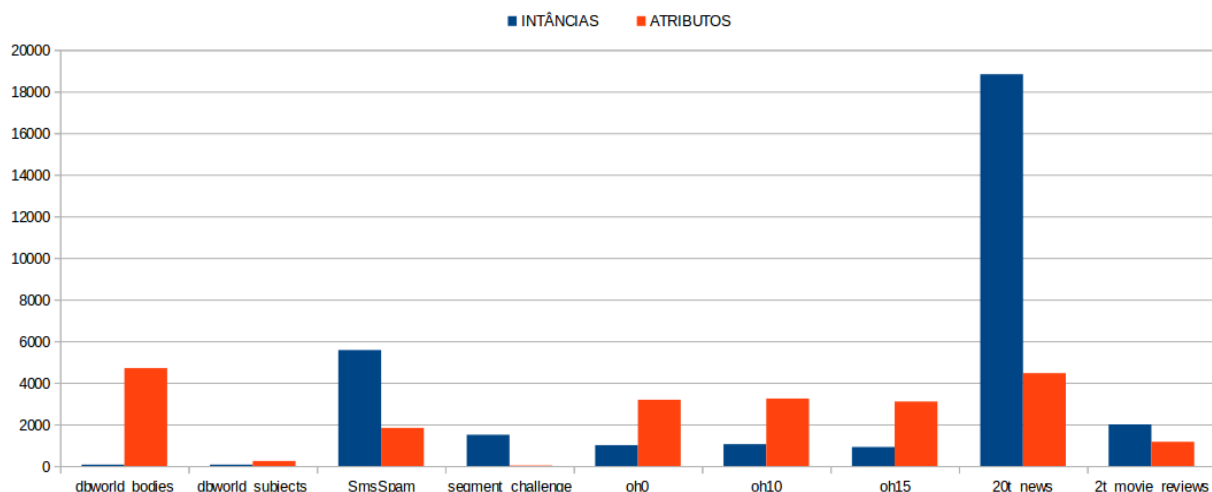


Gráfico 1 - Composição dos conjuntos de dados utilizados
Fonte: Autoria própria

Quanto ao número de instâncias os conjuntos de dados DBWorld_bodies, DBWorld_subjects, SmsSpam, 2t_movieReviews são binários, ou seja, possuem apenas duas classes sem possibilidade de pertencer a ambas durante a classificação, enquanto os demais conjuntos possuem mais de duas classes.

Entre os arquivos utilizados como conjuntos de dados destaca-se a obtenção do conjunto *20 Newsgroups*, composto por textos armazenados em diretórios nomeados conforme as respectivas categorias. Foi utilizado o recurso *TextDirectoryLoader* da interface *SimpleCli* para obtenção de um arquivo com extensão *.arff* a partir de textos separados.

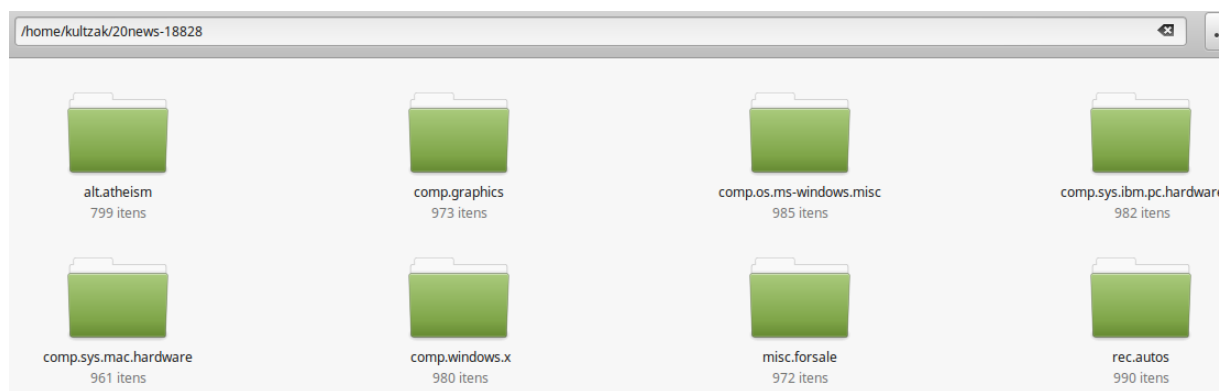
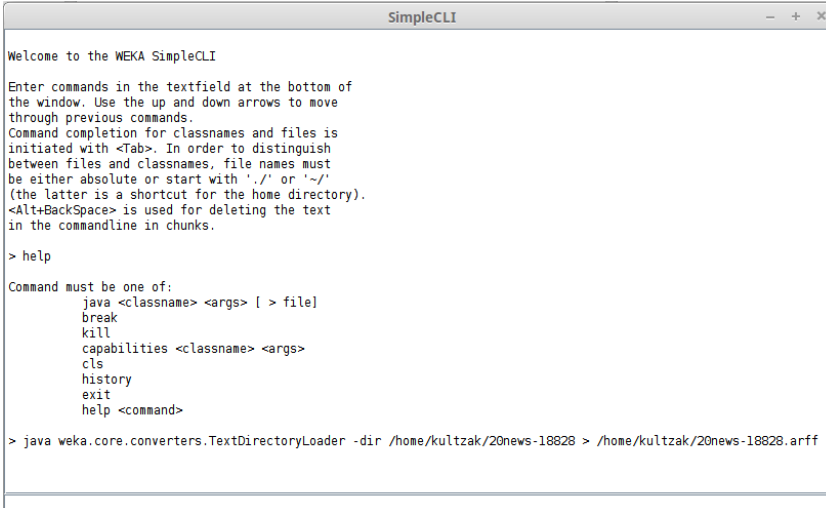


Figura 8 - Estrutura de diretórios que serão convertidos
Fonte: Autoria própria.

Com a aplicação do *TextDirectoryLoader* obtém-se no endereço de destino um arquivo nomeado conforme indicado na linha de comando da Figura 9.



```

SimpleCLI
Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.
Command completion for classnames and files is
initiated with <Tab>. In order to distinguish
between files and classnames, file names must
be either absolute or start with './' or '~/'
(the latter is a shortcut for the home directory).
<Alt+BackSpace> is used for deleting the text
in the commandline in chunks.

> help

Command must be one of:
  java <classname> <args> [ > file]
  break
  kill
  capabilities <classname> <args>
  cls
  history
  exit
  help <command>

> java weka.core.converters.TextDirectoryLoader -dir /home/kultzak/20news-18828 > /home/kultzak/20news-18828.arff

```

Figura 9 - Comando para conversão de diretório em .arff
Fonte: Autoria própria.

3.2 PREPARAÇÃO E PREPROCESSAMENTO

Entre as bases de dados utilizadas neste estudo a única que não apresentava um formato já composto por um vetor de atributos foi a *20 Newsgroups*, logo, foi aplicado o filtro *StringToWordVector* inicialmente respeitando todas as configurações padrão contidas, sem alterar nenhuma configuração.

Os conjuntos de dados *DBWorld_bodies* e *DBWorld_subject* por apresentar apenas 64 instâncias cada em sua forma inicial tiveram que passar por uma reamostragem através do filtro *Resample* e dobraram a quantidade de instâncias, dessa forma os resultados estatísticos podem ser mais significativos, de acordo com implementações de técnicas como *Bootstrap* que se valem de reamostragem em bases de dados com poucos documentos.

Nos casos de seleção de atributos em que foi aplicado o filtro *InfoGain* e *GainRatio* a configuração de corte respeitou o limite de 0, ou seja, todos os atributos que não representassem ganho de informação durante a classificação do algoritmo seriam cortados. No caso da aplicação do filtro *OneR* também para seleção de atributos como os pesos atribuídos possuem diferença em relação às técnicas anteriores, então foi definido seu limite de corte de acordo com a quantidade de

termos para que se igualasse a dos demais filtros de seleção de atributos apresentando assim possibilidade satisfatória de comparação entre si.

A base de dados 20 *Newsgroups* não foi utilizada em sua totalidade (18828 documentos) devido ao tempo de processamento e possível limite de memória alcançado durante a classificação. Com a aplicação do filtro *Resample* foi utilizado 40% da dos documentos originais (7531 documentos).

3.3 SELEÇÃO DE ALGORITMOS

Para efeitos de comparação foram selecionados algoritmos com abordagens diferentes, dessa forma os resultados apresentados durante a classificação serão o cruzamento entre algoritmos com características diferentes e bases de dados também com características diferentes. A seleção abaixo conta com um pequena descrição de cada algoritmo:

- *SMO (Sequential Minimal Optimization)*: implementação do algoritmo SVM substitui todos os valores não encontrados e transforma os atributos nominais em binário, também normaliza todos os atributos por definição.
- *BayesNet*: Rede Bayesiana de aprendizagem que utiliza algoritmos de busca variados e medidas de qualidade.
- *NaiveBayes*: Constrói estimativas dos valores de precisão a partir dos dados usados no treinamento.
- *J48*: Gera uma árvore de decisão implementando o algoritmo C.45.
- *SimpleLogistic*: Constrói um modelo de regressão logística linear simples com o Algoritmo *Logiboost* que define a quantidade ideal de iterações através de *Cross-validation*, o que acarreta seleção automática de atributos.
- *Ibk*: implementação do algoritmo KNN que seleciona o valor de K baseado em *Cross-validation* podendo também atribuir um peso para cada vizinho de K.

3.4 TREINAMENTO E CLASSIFICAÇÃO

Utilizou-se durante a etapa de classificação dois métodos distintos presentes no WEKA, sendo eles, *Cross-validation* e *Percentage Split*.

3.4.1 *Cross-validation*

Durante a classificação através de *Cross-validation* o conjunto de dados é dividido de acordo com o número de dobras (*folds*) especificado, essas dobras correspondem a fração do conjunto que será utilizada para teste, enquanto o restante será aplicado ao treinamento, por exemplo, quando há uma divisão de 10 *folds* então 1/10 desse conjunto será usado na fase de testes e 9/10 para treinamento durante uma iteração em uma sequência de 10 repetições (WEINBERG, 2016).

3.4.2 *Percentage Split*

Quando selecionado o método *Percentage Split* no WEKA deve ser informado o valor relativo a fração da base de dados que será destinada para treinamento e consequentemente o restante será utilizado durante os testes do algoritmo. Quando selecionado através da interface *Experimenter* pode-se designar a preservação da ordem dos documentos ou a seleção aleatória dos mesmos durante a divisão para classificação (WEINBERG, 2016).

Como a interface *Experimenter* do WEKA não conta com a opção de selecionar um subconjunto para treinamento e outro para testes pode-se utilizar o recurso de preservação da ordem do conjunto de dados, dessa forma quando tem-se a divisão entre conjunto de treinamento e conjunto de testes basta que se saiba a proporção que represente cada um e que sejam juntados em um arquivo único, dessa forma como o método não mistura os dados a porcentagem informada será respeitada e as instâncias destinadas a teste e treinamento serão utilizadas de forma respectiva (WEINBERG, 2016).

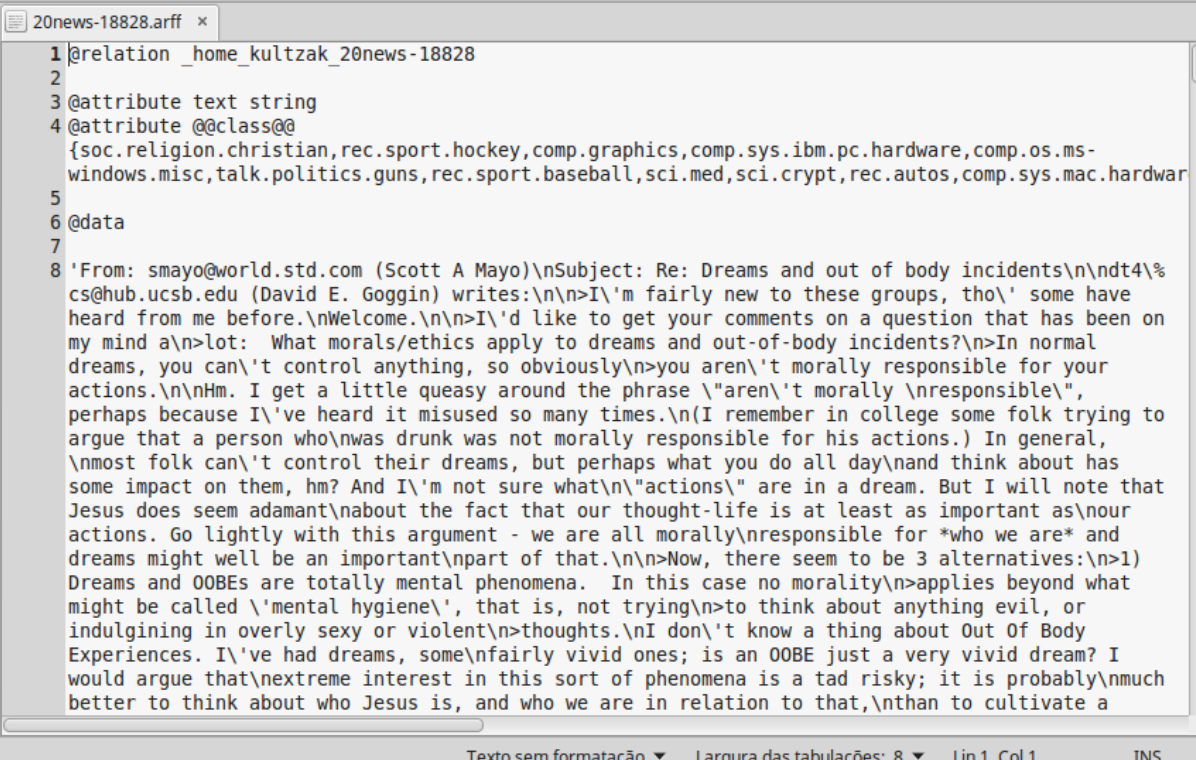
Nenhum dos conjuntos de dados deste trabalho apresenta distinção entre conjunto de treinamento e de testes, sendo assim, durante os testes foram utilizadas as formas de classificação por dados aleatórios e ordem preservada para que pudesse haver maior quantidade de dados durante a avaliação dos resultados.

4 RESULTADOS

As seções subsequentes apresentam os resultados obtidos desde as fases iniciais do processo, iniciando pela etapa de coleta de dados, na sequência apresenta os resultados da fase intermediária de pré-processamento e ao final os números obtidos durante a etapa de classificação.

4.1 RESULTADOS DA COLETA DE DADOS

Pode-se visualizar o resultado da coleta de dados durante o processo de transformação dos arquivos presentes nos diretórios da base 20 *Newsgroups* no arquivo .arff, como foram obtidos em formato simples de texto representam qualquer documento que se tenha intenção de incluir processo de classificação, seja durante a fase de treinamento ou teste.



```

1 @relation _home_kultzak_20news-18828
2
3 @attribute text string
4 @attribute @@class@@
   {soc.religion.christian,rec.sport.hockey,comp.graphics,comp.sys.ibm.pc.hardware,comp.os.ms-
   windows.misc,talk.politics.guns,rec.sport.baseball,sci.med,sci.crypt,rec.autos,comp.sys.mac.hardwar
5
6 @data
7
8 'From: smayo@world.std.com (Scott A Mayo)\nSubject: Re: Dreams and out of body incidents\n\ndt4%\
   cs@hub.ucsb.edu (David E. Goggin) writes:\n\n>I\'m fairly new to these groups, tho\' some have
   heard from me before.\nWelcome.\n\n>I\'d like to get your comments on a question that has been on
   my mind a\n>lot: What morals/ethics apply to dreams and out-of-body incidents?\n>In normal
   dreams, you can\'t control anything, so obviously\n>you aren\'t morally responsible for your
   actions.\n\nHm. I get a little queasy around the phrase \"aren\'t morally \nresponsible\",
   perhaps because I\'ve heard it misused so many times.\n(I remember in college some folk trying to
   argue that a person who\nwas drunk was not morally responsible for his actions.) In general,
   \nmost folk can\'t control their dreams, but perhaps what you do all day\nand think about has
   some impact on them, hm? And I\'m not sure what\n\"actions\" are in a dream. But I will note that
   Jesus does seem adamant\nabout the fact that our thought-life is at least as important as\nour
   actions. Go lightly with this argument - we are all morally\nresponsible for *who we are* and
   dreams might well be an important\npart of that.\n\n>Now, there seem to be 3 alternatives:\n>1)
   Dreams and OOBes are totally mental phenomena. In this case no morality\n>applies beyond what
   might be called \'mental hygiene\', that is, not trying\n>to think about anything evil, or
   indulging in overly sexy or violent\n>thoughts.\nI don\'t know a thing about Out Of Body
   Experiences. I\'ve had dreams, some\nfairly vivid ones; is an OOBE just a very vivid dream? I
   would argue that\nextreme interest in this sort of phenomena is a tad risky; it is probably\nmuch
   better to think about who Jesus is, and who we are in relation to that,\n\nthan to cultivate a

```

Figura 10 - Arquivo .arff obtido após a conversão dos diretórios
 Fonte: Autoria própria.

Observa-se o arquivo criado apresenta dois atributos, o primeiro correspondente a todo o texto no formato *string* e o segundo correspondente às

possíveis classes que cada instância pertence, representadas anteriormente pelo nome dos diretórios que continham os documentos. Na seção de dados pode-se observar os textos transformados em string, sendo que cada linha representa uma instância. Este formato de dados já é suficiente para que a próxima etapa da classificação seja abordada.

4.2 RESULTADOS DO PREPROCESSAMENTO

No caso específico da transformação da base de dados *20 Newsgroups* foram utilizados apenas os valores padrão do filtro *StringToWordVector*, o resultado foi a criação de um arquivo .arff com 5644 termos.

A Figura 10 demonstra através da interface de visualização o conjunto de dados *20 Newsgroups*, o valor 0 representa a não existência do termo no documento e o valor 1 representa a ocorrência.

No.	1: &	2: *	3: +	4: -	5: --	6: --clhj	7: /	8: 00	9: 01	10: 02	11: 03	12: 05	13: 08	14: 09	15: 1	1
	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	N
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
6	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
13	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figura 11 - Visualização da ocorrência dos termos em cada instância
Fonte: Autoria própria.

A quantidade de atributos de cada conjunto de dados original está representada no Gráfico 2, apenas o conjunto de dados *20 Newsgroups* foi excluído

para que não houvesse interferência da sua grande quantidade de instâncias atrapalhando a visualização das demais bases de dados.

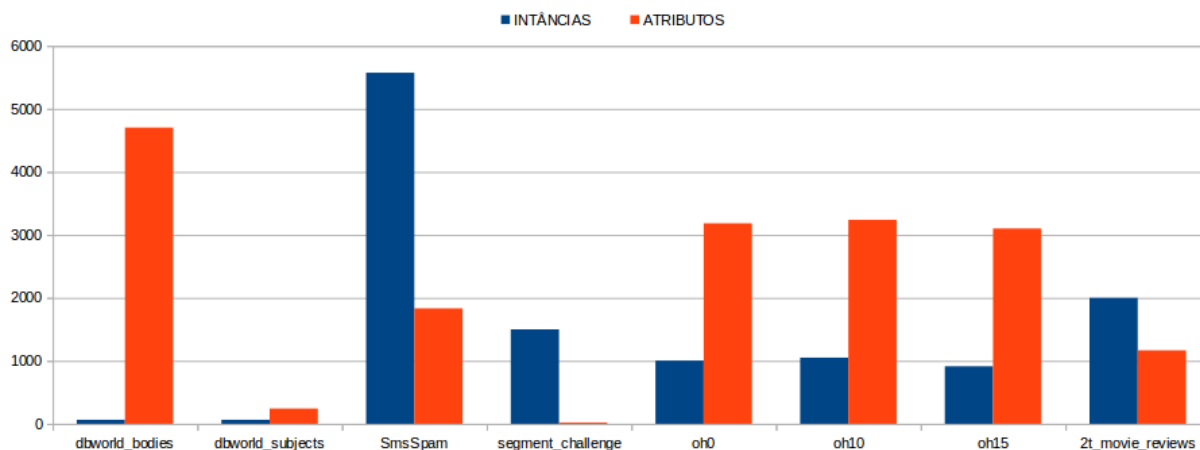


Gráfico 2 - Composição dos conjuntos de dados pré IG/GR/OR
Fonte: Autoria própria

Pode ser feita a comparação com o gráfico 3 o qual representa os resultados da aplicação dos filtros IG (*InfoGain*), GR (*GainRatio*) e OR (*OneR*) para seleção de atributos.

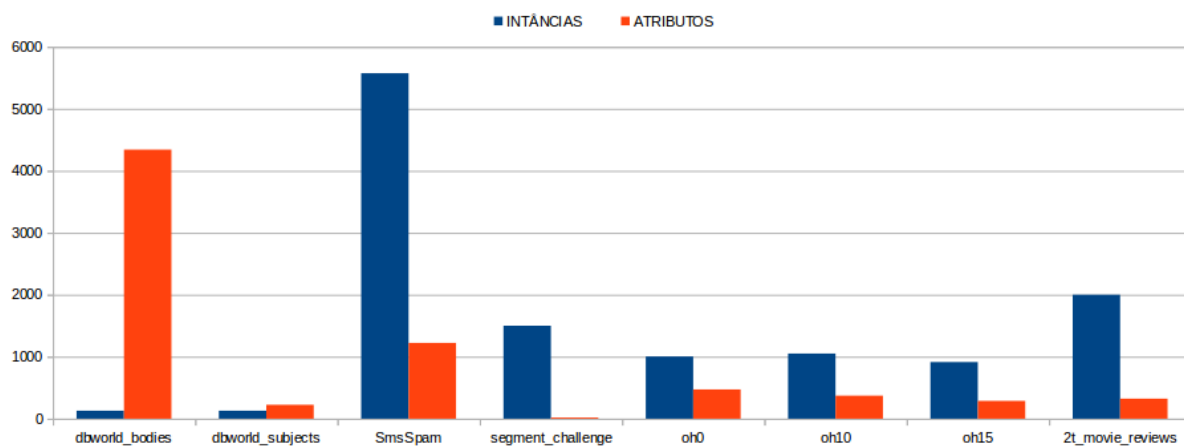


Gráfico 3 - Composição dos conjuntos de dados pós IG/GR/OR
Fonte: Autoria própria

4.3 RESULTADOS DA CLASSIFICAÇÃO

Todos os resultados apresentados nas tabelas correspondem ao percentual de arquivos classificados corretamente.

Os resultados gerados no WEKA foram transcritos para as tabelas incluindo os sinais (*) e (v) que correspondem respectivamente a indicação de pior e melhor em relação ao campo base, como forma de padronização o algoritmo SMO foi definido como base para todos os testes.

Para chegar às conclusões de pior e melhor resultado o WEKA utiliza a comparação através do Paired T-test que é um teste estatístico utilizado para avaliar amostras nas quais as observações podem ser comparadas com outras avaliações, como nos casos de antes e depois ou comparativo de dois métodos de medida ou dois tratamentos (SHIER, 2004).

Testou-se através do *Experimenter* a base de dados *20 Newsgroups* que após seu tratamento acabou pronta para a tarefa de classificação com 40% das instâncias originais, o método utilizado foi o de 10 *folds Cross-validation* e os resultados podem ser conferidos na Figura 11.

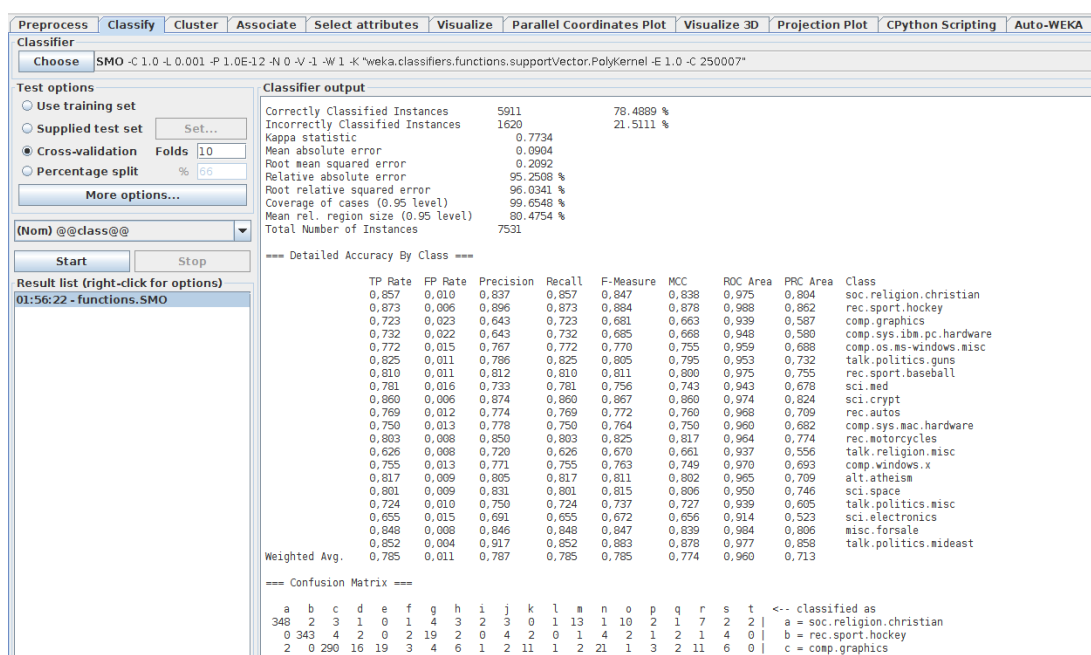


Figura 12 - Tela de resultado da classificação da base de dados *20 Newsgroups*
Fonte: Autoria própria

O WEKA disponibiliza os resultados no formato de texto plano e pode-se observar a quantidade de arquivos classificados corretamente além de outras estatísticas e as principais medidas avaliação para cada classe são exibidas em uma tabela, pode-se observar também uma matriz de confusão detalhada na Figura 12.

A matriz de confusão gerada possibilita uma visualização da quantidade de instâncias classificadas corretamente e incorretamente em cada classe, em alguns casos a matriz de confusão facilita a observação de determinadas características da base de dados, no caso desta pode-se verificar que entre os documentos que pertencem à classe soc.religion.christian e foram classificados incorretamente na primeira e segunda posição em números de FP (Falso Positivo) estão as classes talk.religion.mist e alt.atheism que possuem conteúdo semelhante a classe correta, levando a uma maior tendência de FP para tais classes do que nas demais.

```

=== Confusion Matrix ===
 a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  <-- classified as
348 2  3  1  0  1  4  3  2  3  0  1 13  1 10  2  1  7  2  2 | a = soc.religion.christian
 0 343 4  2  0  2 19  2  0  4  2  0  1  4  2  1  2  1  4  0 | b = rec.sport.hockey
 2  0 290 16 19  3  4  6  1  2 11  1  2 21  1  3  2 11  6  0 | c = comp.graphics
 0  1 18 281 26  1  1  3  1  4 14  1  0  7  1  1  0 16  8  0 | d = comp.sys.ibm.pc.hardware
 1  0 22  29 339  1  3  3  4  2 10  2  1 12  1  1  0  4  4  0 | e = comp.os.ms-windows.misc
 3  2  1  1  0 293  3  7  6  2  3  2  4  2  1  4 14  0  1  6 | f = talk.politics.guns
 5 19  4  6  1  4 336  4  1  4  1  4  2  5  2  3  7  3  2  2 | g = rec.sport.baseball
 3  1  8  5  1  4  7 304  2  8  7  2  3  5  0  9  8  7  4  1 | h = sci.med
 1  1  6  2  2  9  2  6 313  2  1  2  0  6  1  1  2  6  0  1 | i = sci.crypt
 1  0  9  2  4  2  7  5  1 287  3 15  0  3  3  6  4 11  8  2 | j = rec.autos
 1  1 15 28 16  1  2  9  1  8 312  1  0  5  0  1  0  9  4  2 | k = comp.sys.mac.hardware
 2  2  2  4  1  4  5  9  4 14  2 305  1  4  1  7  2  4  6  1 | l = rec.motorcycles
24  1  2  1  1  9  1  6  2  3  1  1 152  1 27  3  6  0  0  2 | m = talk.religion.misc
 1  4 34 13 17  0  0  6  1  1  6  2  1 320  0  2  2 11  3  0 | n = comp.windows.x
10  1  4  3  1  2  3  6  1  4  0  1 17  0 281  1  5  0  0  4 | o = alt.atheism
 4  0  3  5  3  2  3 15  3  7  4  3  4  3  3 309  7  6  1  1 | p = sci.space
 2  1  0  2  0 28  6  5  8  1  0  3  2  2  9  6 213  1  0  5 | q = talk.politics.misc
 1  3 16 27  6  4  2  8  5 10 17  7  2  6  1  5  1 239  5  0 | r = sci.electronics
 0  0  7  9  5  1  5  2  1  5  7  3  0  6  0  2  0  5 324  0 | s = misc.forsale
 7  1  3  0  0  2  1  6  1  0  0  3  6  2  5  5  8  5  1 322 | t = talk.politics.mideast

```

Figura 13 - Matriz de confusão gerada pelo algoritmo de aprendizagem
Fonte: Autoria própria

Para observações mais detalhadas da classificação de textos pode-se ativar através das opções o recurso que gera as predições realizadas pelo WEKA como visto na Figura 13.

The screenshot shows the Weka Explorer interface. The 'Classifier' tab is active, displaying the SMO classifier. The 'Test options' section shows 'Cross-validation' selected with 10 folds. The 'Classifier output' pane displays the following information:

```

=== Run information ===
Scheme: weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K *weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007*
Relation: home_kultzak_20news-18828-weka.filters.unsupervised.attribute.StringToWordVector-R1-W1000-prune-rate-1.0-NO-stemmerweka.cor
Instances: 7531
Attributes: 4462
Test mode: 10-fold cross-validation

=== Predictions on test data ===
inst#  actual  predicted error prediction
 1 3:comp.graphics 3:comp.graphics 0.1
 2 3:comp.graphics 3:comp.graphics 0.1
 3 3:comp.graphics 3:comp.graphics 0.1
 4 3:comp.graphics 3:comp.graphics 0.1
 5 3:comp.graphics 3:comp.graphics 0.1
 6 3:comp.graphics 4:comp.sys.ibm.pc.hardware + 0.1
 7 3:comp.graphics 9:sci.crypt + 0.1
 8 3:comp.graphics 14:comp.windows.x + 0.095
 9 3:comp.graphics 3:comp.graphics 0.1
10 3:comp.graphics 3:comp.graphics 0.1
11 3:comp.graphics 3:comp.graphics 0.1
12 3:comp.graphics 11:comp.sys.mac.hardware + 0.1
13 3:comp.graphics 3:comp.graphics 0.1
14 3:comp.graphics 3:comp.graphics 0.1
15 3:comp.graphics 5:comp.os.ms-windows.misc + 0.095
16 3:comp.graphics 3:comp.graphics 0.1
17 3:comp.graphics 3:comp.graphics 0.095
18 3:comp.graphics 3:comp.graphics 0.1
19 3:comp.graphics 3:comp.graphics 0.1
20 3:comp.graphics 11:comp.sys.mac.hardware + 0.1
21 3:comp.graphics 3:comp.graphics 0.1
22 3:comp.graphics 3:comp.graphics 0.095
23 3:comp.graphics 3:comp.graphics 0.1
24 3:comp.graphics 3:comp.graphics 0.1

```

Figura 14 - Predições do algoritmo por instância
Fonte: Autoria própria

No caso da Figura 13 os resultados foram gerados em texto plano mas as configurações possibilitam que as predições sejam armazenadas em outros formatos como o CSV, assim é possível que sejam rastreados os arquivos caso a intenção da tarefa de classificação seja prever a classe pertencente de um documento.

A fim de verificar os resultados da amostragem para bases de dados com poucos atributos foi realizado um teste com a base de dados DBWorld_bodies:

Tabela 2 - Conjuntos particionados com Percentage Split 66% dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
dbWorld_bodies Original	85,71	80,95	71,43	66,67	76,19	57,14
dbWorld_bodies Resample	97,67	88,37	81,40	88,37	88,37	83,72

Fonte: Autoria própria

Nota-se uma melhora em todos os resultados após a aplicação do filtro *Resample* gerando o dobro de instâncias para classificação.

Ainda na base de dados DBWorld_bodies, utilizando o filtro IG foram retirados todos os *tokens* que não apresentaram ganho de informação e o restante foi particionado em 10 conjuntos de dados que representam respectivamente de 10 a 100% dos termos mais relevantes de acordo com posição no ranking, sendo assim o conjunto de dados menor com apenas 10% dos termos é o composto por aqueles com maior peso.

Tabela 3 - Conjuntos particionados com Percentage Split 66% dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
dbWorld_bodies 10	97,67	86,05*	86,05*	86,05*	88,37*	88,37*
dbWorld_bodies 20	97,67	86,05*	83,72*	86,05*	88,37*	86,05*
dbWorld_bodies 30	97,67	86,05*	83,72*	86,05*	93,02*	76,74*
dbWorld_bodies 40	97,67	86,05*	79,07*	86,05*	93,02*	76,74*
dbWorld_bodies 50	97,67	90,70*	81,40*	86,05*	90,70*	79,07*
dbWorld_bodies 60	97,67	88,37*	83,72*	86,05*	90,70*	88,37*
dbWorld_bodies 70	97,67	90,70*	81,40*	86,05*	90,70*	83,72*
dbWorld_bodies 80	97,67	90,70*	81,40*	86,05*	90,70*	83,72*
dbWorld_bodies 90	97,67	88,37*	81,40*	86,05*	88,37*	83,72*
dbWorld_bodies 100	97,67	88,37*	81,40*	88,37*	88,37*	83,72*

Fonte: Autoria própria

Tabela 4 - Conjuntos particionados com *Percentage Split* 66% ordem preservada

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
dbWorld_bodies 10	90,91	88,64*	93,18v	88,64*	90,91	86,36*
dbWorld_bodies 20	90,91	93,18*	95,45v	90,91	90,91	90,91
dbWorld_bodies 30	90,91	95,45v	95,45v	90,91	90,91	88,64*
dbWorld_bodies 40	90,91	95,45v	95,45v	90,91	90,91	88,64*
dbWorld_bodies 50	90,91	95,45v	95,45v	90,91	90,91	86,36*
dbWorld_bodies 60	90,91	95,45v	95,45v	90,91	90,91	86,36*
dbWorld_bodies 70	90,91	95,45v	95,45v	90,91	90,91	88,64*
dbWorld_bodies 80	90,91	95,45v	95,45v	90,91	90,91	88,64*
dbWorld_bodies 90	90,91	95,45v	95,45v	90,91	90,91	88,64*
dbWorld_bodies 100	90,91	90,91	95,45v	90,91	90,91	88,64*

Fonte: Autoria própria

Tabela 5 - Conjuntos particionados com 10 fold *Cross-validation*

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
dbWorld_bodies 10	95,65	90,36	88,99	91,73	91,52	89,01
dbWorld_bodies 20	95,88	90,59	88,52*	91,96	93,11	90,36
dbWorld_bodies 30	95,65	91,51	88,97	91,96	92,67	87,16
dbWorld_bodies 40	95,65	91,96	89,20	91,96	93,35	86,24
dbWorld_bodies 50	95,88	92,65	89,66	92,19	93,57	86,93
dbWorld_bodies 60	95,88	91,74	89,20	92,19	92,89	89,46
dbWorld_bodies 70	95,41	91,96	89,20	92,19	92,44	88,08
dbWorld_bodies 80	95,88	91,73	87,60	92,19	92,44	88,08
dbWorld_bodies 90	95,41	91,50	87,83	92,19	92,66	90,13
dbWorld_bodies 100	95,41	88,53	86,01	91,74	91,52	87,62

Fonte: Autoria própria

Observa-se que no caso da terceira tabela que utiliza a classificação através de *Cross-validation* os valores dos resultados apresentaram proximidade suficiente para que não houvesse distinção entre qual algoritmo teve melhor aproveitamento.

A diferença entre a quantidade de termos selecionados, que foi objetivo deste teste, não foi relevante para o algoritmo SMO, enquanto para outros algoritmos dependendo da quantidade escolhida há uma pequena vantagem sobre a utilização da totalidade dos atributos.

Entre a forma de classificação utilizada houve diferença entre a que utiliza dados aleatórios e a que mantém a ordem dos dados preservada, no caso da

primeira forma o algoritmo SMO se mostrou superior a todos dos demais enquanto na segunda os algoritmos bayesianos se comportaram melhor.

Para a verificação de outra etapa da fase de pré-processamento dos textos foi aplicado durante a transformação *StringToWordVector* da base de dados 20 *NewsGroups* uma lista *stopwords* (GANESAN, 2016).

Tabela 6 - Resultados *InfoGain* com *Percentage Split* 66% dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
20_newsGroups	74,17	69,05*	68,06v	66,94*	81,95v	52,40*
20_newsGroups <i>stopwords</i>	77,41	73,43*	75,22*	67,96*	81,71*	51,58*

Fonte: Autoria própria

Com exceção dos algoritmos SimpleLogistic e Ibk os demais apresentaram melhor resultado no conjunto de dados em que foi utilizada a lista de *stopwords*. Ainda na base de dados 20_newsGroups foram aplicados os filtros *GainRatio* e *InfoGain* para seleção de atributos, enquanto em outra cópia da base de dados foi feita a transformação de string para vetor de termos habilitando-se o *stemming*, a contagem de termos por documento, além de *IDFTransform* e *TFTransform*.

Tabela 7 - Resultados 20 *NewsGroups* dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
20_newsGroups GR	74,48	69,05*	66,78*	67,72*	81,52v	52,13*
20_newsGroups IG	74,48	69,05*	66,78*	66,43*	81,52v	52,13*
20_newsGroups C/	80,70	68,11*	69,64*	65,46*	80,66*	60,22*

Fonte: Autoria própria

Como durante a transformação *StringToWordVector* obteve-se um número menor de atributos foi realizado um novo teste quem que as duas primeira amostras tiveram a quantidade de atributos igualadas a terceira através da reaplicação GR e IG com seleção manual da quantidade de atributos.

Tabela 8 - Resultados 20 *NewsGroups* atributos igualados

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
20_newsGroups GR	75,07	71,36*	72,84*	64,20*	77,61v	71,94*
20_newsGroups IG	70,18	66,39*	64,99*	66,98*	77,73v	50,18*
20_newsGroups C/	80,70	68,11*	69,64*	65,46*	80,66*	60,22*

Fonte: Autoria própria

As tabelas 6 e 7 apresentaram melhores resultados para o conjunto de dados em que a etapa de transformação de *string* para vetor de termos foi feita com os recursos de *stemming* e determinação da frequência de termos ativa independente de não ter sido feita seleção de atributos na mesma.

Os próximos testes apresentados envolveram um comparativo entre os 8 conjuntos de dados e os 6 algoritmos de classificação utilizando as três abordagens disponíveis na interface *Experimenter*. Os *tokens* que não apresentassem ganho de informação foram retirados através do filtro de seleção de atributos *InfoGain*.

Tabela 9 - Resultados *InfoGain* com *Percentage Split* 66% dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	81,03	75,88*	85,00v	65,00*	82,94v	57,79*
dbWorld_bodies	97,67	88,37*	81,40*	86,05*	88,37*	83,72*
dbWorld_subject	90,70	90,70	90,70	81,40*	86,05*	90,70
oh0	82,80	85,42v	80,76*	79,30*	82,80	67,35*
oh10	77,53	77,25*	75,00*	77,25*	78,65v	52,53*
oh15	80,71	79,10*	72,99*	76,85*	82,96v	62,06*
segment_challenge	90,98	89,02*	81,57*	96,47v	94,71v	96,08v
smsSpam	98,73	98,31*	96,94*	94,78*	98,47*	94,46*

Fonte: Autoria própria

Na tabela 9 é possível notar que a base de dados smsSpam apresentou um resultado baixo em relação aos demais na própria tabela e em relação às abordagens da tabela 8 e 10.

Tabela 10 - Resultados *InfoGain* com *Percentage Split* 66% ordem preservada

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	84,26	78,68*	86,47v	71,03*	80,88*	60,59*
dbWorld_bodies	90,91	90,91	95,45v	90,91	90,91	88,64*
dbWorld_subject	90,91	88,64*	88,64*	86,36*	86,36*	88,64*
oh0	78,89	86,51v	82,11v	78,30*	84,16v	62,76*
oh10	70,31	72,27v	74,23v	72,55v	76,19v	45,94*
oh15	77,10	73,87*	75,48*	74,19*	82,90*	56,77*
segment_challenge	89,61	91,18v	80,00*	93,53v	93,92v	94,12v
smsSpam	60,58	60,58	60,58	60,58	60,58	60,58

Fonte: Autoria própria

Tabela 11 - Resultados *InfoGain* com 10 fold Cross-validation

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	84,85	80,45*	84,85	67,80*	85,10	58,80*
dbWorld_bodies	97,69	91,47	88,40*	93,01	91,47	89,10*
dbWorld_subject	98,46	96,15	96,15	93,08*	96,15	96,92
oh0	85,94	89,43v	84,74	83,45	87,93	70,20*
oh10	78,86	80,67	77,62	79,43	79,52	56,95*
oh15	90,93	82,90	80,28	79,51	83,79	65,17*
segment_challenge	91,87	90,40	81,13*	95,67v	95,13v	96,80v
smsSpam	98,47	92,24	96,93	96,09*	98,21	95,42*

Fonte: Autoria própria

A tabela 10 possui alguns casos em que a variância entre os dados acabou não produzindo diferenciação entre os resultados dos algoritmos como no caso da base oh15 em que só foi possível identificar o pior resultado como sendo IBk por estar bem abaixo dos demais.

Comparativo entre os 8 conjuntos de dados e os 6 algoritmos de classificação. Seleção de atributos com o filtro *GainRatio*.

Tabela 12 - Resultados *GainRatio* com dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	81,03	75,88*	85,00v	65,29*	82,94v	57,79*
dbWorld_bodies	97,67	88,37*	81,40*	90,70*	88,37*	83,72*
dbWorld_subject	100	97,67*	97,67*	90,70*	100	97,67*
oh0	83,09	85,42v	80,76*	80,17*	82,80*	67,35*
oh10	77,53	77,25*	75,00*	77,25*	78,65v	53,53*
oh15	80,71	79,10*	72,99*	77,17*	82,96v	62,06*
segment_challenge	90,98	89,02*	81,57*	96,47v	94,71v	96,08v
smsSpam	98,15	98,21v	96,78*	95,46*	97,89*	94,88*

Fonte: Autoria própria

Tabela 13 - Resultados *GainRatio* com ordem preservada

(continua)

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	84,26	78,68*	86,47v	70,44*	80,88*	60,59*
dbWorld_bodies	90,91	90,91	95,45v	90,91	90,91	88,64*
dbWorld_subject	77,27	79,55v	43,18*	29,55*	84,36v	75,00*

Tabela 13 - Resultados *GainRatio* com ordem preservada**(conclusão)**

Dataset	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
oh0	78,89	86,51v	82,11v	78,59*	84,16v	62,76*
oh10	70,31	72,27v	74,23v	72,83v	76,19v	45,95*
oh15	77,42	73,87*	75,48*	74,19*	72,90*	56,77*
segment_challenge	89,61	91,18v	80,00*	93,53v	93,92v	94,12v
smsSpam	98,21	98,05*	96,94*	95,41*	97,76*	94,25*

Fonte: Autoria própria

Nas tabelas 11, 12 e 13 encontra-se os resultados da aplicação além do *InfoGain* que calcula o valor dos atributos, do filtro *SubsetEval*, que é uma forma de avaliar quais são os subconjuntos mais adequados para a classificação.

Tabela 14 - Resultados *GainRatio* com *Cross-validation*

Algoritmo						
Dataset	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	84,85	80,45*	84,85	67,85*	85,10	58,80*
dbWorld_bodies	97,69	91,47	88,40*	93,01	91,47	89,10*
dbWorld_subject	99,23	96,79	96,79	92,95	96,86	95,26
oh0	85,94	89,43v	84,74	83,55	87,93	70,20*
oh10	78,86	80,67	77,62	79,71	79,52	56,95*
oh15	81,04	82,90	80,28*	79,95	83,79	65,17*
segment_challenge	91,87	90,40	81,13	95,67v	95,13v	96,80v
smsSpam	98,39	98,19	96,93	96,20*	98,24	95,16*

Fonte: Autoria própria**Tabela 15 - Resultados *InfoGain SubsetEval* com dados aleatórios**

Algoritmo						
Dataset	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	80,88	75,88*	82,35v	71,91*	80,29*	66,03*
dbWorld_bodies	97,67	95,35*	95,35*	93,02*	95,35*	93,02*
dbWorld_subject	93,02	93,02	93,02	90,70*	93,02	93,02
oh0	67,93	81,92v	75,22v	81,63v	82,51v	73,47v
oh10	67,42	76,97v	70,79v	75,84v	75,84v	65,17*
oh15	68,81	77,49v	70,10v	75,24v	78,46v	63,67*
segment_challenge	86,67	94,12v	83,53*	96,67v	91,96v	92,55v
smsSpam	96,67	97,68v	95,46*	93,98*	97,52v	96,41*

Fonte: Autoria própria

Analisando os resultados das tabelas 14, 15 e 16 nota-se um aumento da eficiência após a aplicação do filtro *SubsetEval* na base de dados DBWorld_subject, nas bases de dados oh0, oh10 e oh15 houve melhora em relação ao algoritmo IBk e piora entre os demais.

Tabela 16 - Resultados InfoGain SubsetEval com ordem preservada

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	81,62	77,79*	83,24v	72,94*	81,32*	64,56*
dbWorld_bodies	95,45	93,18*	93,18*	93,18*	95,45	95,45*
dbWorld_subject	84,09	93,18v	93,18v	79,55*	84,09	84,09
oh0	64,52	82,99v	76,54v	80,65v	81,03v	70,38v
oh10	54,34	73,95v	70,31v	74,51v	72,27v	62,18v
oh15	60,00	76,45v	69,68v	67,42v	77,10v	60,65v
segment_challenge	85,29	90,59v	78,63*	91,18v	92,35v	94,71v
smsSpam	60,58	60,58	60,58	60,58	60,58	60,58

Fonte: Autoria própria

Outro teste com seleção de atributos foi implementado, utilizando o Filtro OneR que apresenta pesos diferentes dos atribuídos por IG e GR, para se manter a linha de testes o número de instâncias foi escolhido manualmente utilizando o ranking do filtro em número igual as apresentadas nos testes com IG e GR anteriormente.

Tabela 17 - Resultados InfoGain SubsetEval com Cross-validation

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	81,80	79,45*	81,60	73,35*	81,80	66,60*
dbWorld_bodies	98,46	92,24	92,24	90,64*	98,46	98,46
dbWorld_subject	91,35	91,35	91,35	86,67*	89,81	91,35
oh0	70,00	85,65v	76,78v	83,65v	84,74v	76,07v
oh10	67,05	76,76v	68,67	75,62v	75,33v	65,05
oh15	70,00	80,07v	72,07	76,01v	80,06v	68,11
segment_challenge	87,07	92,80v	81,73*	95,33v	93,47v	95,07v
smsSpam	96,88	97,09	96,05*	95,80*	97,41v	96,52

Fonte: Autoria própria

A Tabela 19 apresenta o Filtro *OneR* com quantidade de atributos selecionada de forma manual.

Tabela 18 - Resultados *OneR* com dados aleatórios

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	81,03	75,88*	85,00v	65,29*	82,94*	57,79*
dbWorld_bodies	97,67	88,37*	81,40*	86,05*	95,35*	83,72*
dbWorld_subject	100	97,67*	97,67*	90,70*	100	97,67*
oh0	79,59	86,01v	78,72*	79,88v	84,84v	48,40*
oh10	75,28	77,81v	71,35*	76,97v	76,69v	46,07*
oh15	74,28	78,46v	70,42*	76,53v	79,74v	42,12*
segment_challenge	90,98	89,02*	81,57*	96,47v	94,71v	96,08v
smsSpam	98,15	97,84*	95,78*	94,78*	97,57*	94,72*

Fonte: Autoria própria

Tabela 19 - Resultados *OneR* com ordem preservada

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	84,26	78,68*	86,47v	71,03*	80,88*	60,59*
dbWorld_bodies	90,91	95,45v	95,45v	90,91	90,91	88,64*
dbWorld_subject	77,27	79,55v	38,64*	29,55*	86,36v	75,00*
oh0	76,83	86,22v	80,65v	80,06v	85,63v	46,33*
oh10	68,35	73,67v	68,35	70,87v	73,95v	33,33*
oh15	70,32	72,90v	72,90v	73,23v	74,19v	35,81*
segment_challenge	89,61	91,18v	80,00*	93,53v	93,92v	94,12v
smsSpam	98,05	98,10v	96,25*	95,30*	97,47*	93,93*

Fonte: Autoria própria

Para comparação da eficiência dos seis classificadores escolhidos foi elaborada a tabela 20 que os posiciona de acordo com o número de posições conquistadas durante os testes realizados.

Tabela 20 - Resultados *OneR* com *Cross-validation*

Dataset	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
2t_movie_reviews	84,90	80,45*	84,85	67,40*	85,10	58,80*
dbWorld_bodies	97,69	92,24*	88,40*	92,24	94,55	89,94*
dbWorld_subject	99,23	96,79	98,46	92,95	98,46	95,26
oh0	83,05	90,13v	81,25	82,85	86,33	51,85*
oh10	77,90	80,48	72,10*	75,05	78,10	47,14*
oh15	75,03	81,14v	74,59	76,13	79,19	48,21*
segment_challenge	91,87	90,40	81,13*	95,67v	95,13v	96,80v
smsSpam	98,44	98,17	96,61*	95,05*	98,13	94,99*

Fonte: Autoria própria

Não foram incluídos na tabela os resultados obtidos através de *Cross_validation*, pois a variância obtida impediu a maioria dos resultados de apresentar diferenças significativas na comparação entre os algoritmos. Também foram excluídos os testes das tabelas 2 e 3 que apresentavam muita similaridade entre os resultados para uma única base de dados contribuindo para poluir os resultados da tabela 20, e por último foram excluídos os resultados em que todos os algoritmos tiveram a mesma pontuação por não influenciar o resultado final.

Tabela 21 - Número de vezes em cada posição por algoritmo

Posição	Algoritmo					
	SMO	BayesNet	NaiveBayes	J48	SimpleLogistic	IBk
1º	18	16	13	7	25	6
2º	22	19	10	6	21	8
3º	7	11	10	22	19	3
4º	11	20	14	8	4	5
5º	11	5	11	25	1	9
6º	2	0	13	4	1	40

Fonte: Autoria própria

O algoritmo de Classificação que teve o maior número de primeiras colocações foi o *SimpleLogisti* seguido por SMO e *BayesNet*. O algoritmo IBk ficou o maior número de vezes na última colocação.

De acordo com a base de dados utilizada foi elaborado um gráfico para observar quais algoritmos tiveram destaque durante sua classificação:

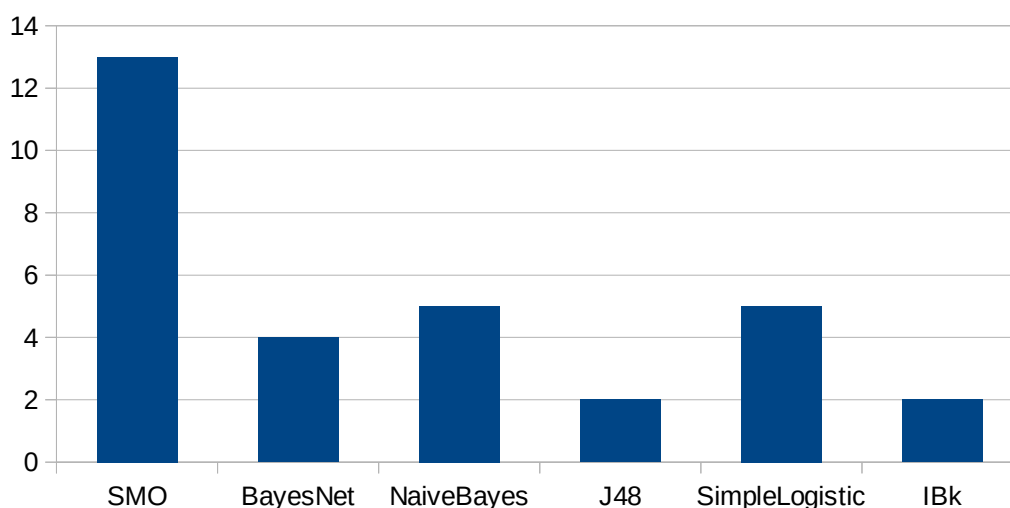


Gráfico 4 - Número de 1ª colocações nas bases de dados DB_World

Fonte: Autoria própria

O algoritmo SMO apresentou melhor desempenho quando aplicado as bases de dados DB_World.

A base de dados 2T_movieReviews teve a 1ª colocação durante os testes realizados alcançada sempre pelo algoritmo *NaiveBayes*.

O algoritmo *SimpleLogistic* obteve melhor desempenho na base de dados 20 *Newsgroups* em 6 dos 7 testes e também nas bases de dados OH como observado no gráfico 5.

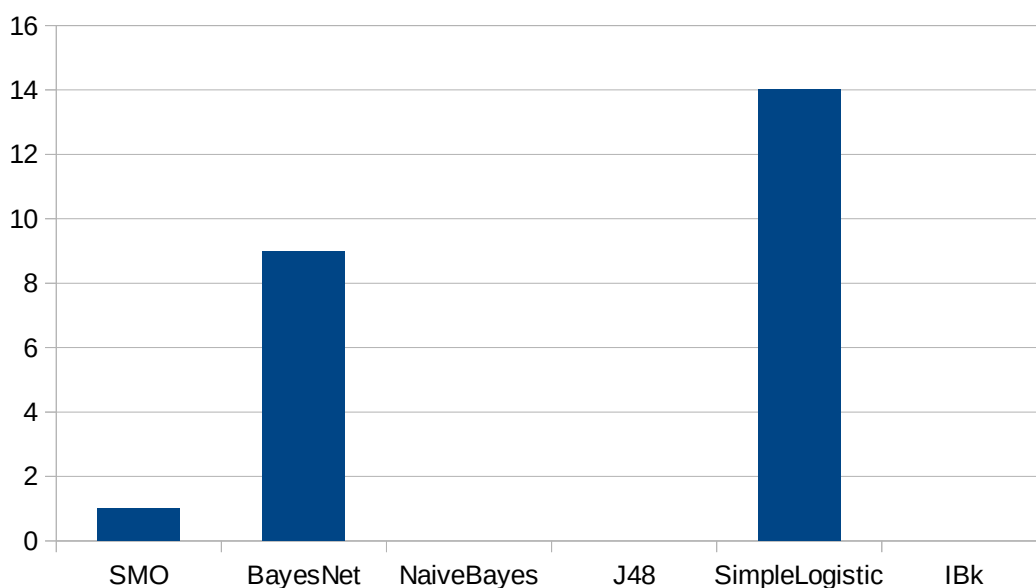


Gráfico 5 - Número de 1ª colocações nas bases de dados OH
Fonte: Autoria própria

Embora os algoritmos IBk e J48 tenham obtido piores desempenhos no geral durante a classificação dos conjuntos de dados SegmentChallenge foram os melhores colocados com 4 primeiras colocações cada. A base de dados smsSpam teve as primeiras posições divididas entre SMO e BayesNet com 3 primeiras posições para cada um nos testes utilizados como parâmetro.

5 CONCLUSÃO

Dentro do objetivo deste trabalho de discorrer quanto a classificação de textos utilizando algoritmos de aprendizagem de máquina com WEKA pôde ser demonstrado através do processo completo desde a obtenção dos dados até a avaliação dos resultados como algumas técnicas atuais podem ser utilizadas para se obter a separação de documentos em classes utilizando o processo de aprendizagem supervisionada.

Quanto à aplicação de técnicas que buscam obter melhores resultados na atividade de classificação é notável a influência de uma adequada fase de pré-processamento utilizando recursos como reamostragem nos casos de baixa quantidade de instâncias, porém pode ser observado também que em alguns casos a seleção de atributos não afeta os resultados de maneira positiva, podendo manter os valores apresentados antes da seleção ou até mesmo obter resultados piores.

Gabrilovich e Markovitch (2004) tratam dos resultados da seleção de atributos inclusive com a possibilidade de quantificação. Acessando os valores de IG e os *outliers*, valores que estão bem acima da média dos outros termos. A base de dados 20 *Newsgroups* conta com valor de *outliers* de 391, o que pode justificar os resultados não melhorados após a seleção de atributos.

Quanto aos classificadores selecionados obteve-se um melhor desempenho daquele que apresentava o recurso de seleção de atributos embutido, no caso *SimpleLogistic*, a maior regularidade pôde ser observada com o *NaiveBayes* que entre os resultados de 1ª a 6ª colocações variou no máximo em 4 pontos, e o pior desempenho foi apresentado pelo IBk que teve o maior número de últimas colocações entre os avaliados.

Quanto a identificação dos padrões nada é conclusivo porém observou-se que:

1. O algoritmo possuidor de seleção de atributos embutido obteve melhor resultado naquelas bases de dados com maior quantidade de atributos originalmente.
2. O algoritmo SMO apresentou maior afinidade com as bases de dados possuidoras dos extremos de menor e maior número de instâncias em relação aos atributos.

3. Embora J48 e IBk tenham apresentado resultados inferiores quando observado a totalidade dos teste, nas bases de dados com o menor número de atributos foram os que apresentaram o melhor desempenho.

Relacionado com as peculiaridades da técnica em si, pode ser considerada a grande dimensionalidade das bases de dados um fator dificultante para a classificação, tendo que vista que a tarefa de pré-processamento acaba sendo bastante requisitada para a obtenção de subconjuntos das bases de dados satisfatórios durante os testes tanto pelos algoritmos não serem eficientes com muitos termos quanto pelo custo computacional requisitado para muitas instâncias.

5.1 TRABALHOS FUTUROS

Trabalhos futuros podem incluir uma maior pesquisa sobre bases de dados em português e estudo específico da seleção de atributos na língua portuguesa, tendo em vista que a aplicação de *stopwords* e outros recursos são específicos para determinada língua e suas peculiaridades sintáticas e morfológicas.

Como determinado classificadores acabam se saindo melhor com alguns tipos de base de dados, poderia ser elaborado um método de definição automática de melhor classificador de acordo com o tipo de texto baseado em metadados que integrariam os conjuntos e informações mais detalhadas dos classificadores, que poderia trabalhar em conjunto com técnicas como EM (NIGAM et al., 2000).

Os avanços da linguística podem abrir espaço para a construção de um modelo universal e eficiente para todos os casos, em trabalhos futuros pretende-se documentar as características específicas dos classificadores e etapas de pré-processamento para subsidiar a atribuição automática de classes.

REFERÊNCIAS

AGGARWAL, Charu C; ZHAI, ChengXiang,. **A survey of text classification algorithms**. Mining text data. Springer US, p. 163-222, 2012.

AHA, David W; KIBLER, Dennis; ALBERT, Marc K. **Instance-based learning algorithms**. Machine learning 6.1, p. 37-66, 1991.

BLUM, Avrim L; LANGLEY, Pat. **Selection of relevant features and examples in machine learning**. Artificial intelligence 97.1, p. 245-271, 1997.

WEINBERG, Brandon. **WEKA Text Classification for First Time & Beginner Users**. Disponível em: <<http://www.youtube.com/watch?v=M3oLEGlzs6k>>. Acesso em: 5 maio. 2016.

BROWNLEE, Jason. **“Practical Machine Learning Problems”**. Disponível em: <<http://machinelearningmastery.com/practical-machine-learning-problems/>>. Acesso em: 14 nov 2015.

CARRILHO, João Ribeiro. **Desenvolvimento De Uma Metodologia Para Mineração De Textos**. Disponível em: <http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=11675@1>. Acesso em: 12 jan 2016.

CHOMSKY, Noam. **Language and freedom**. Resonance 4.3, p. 86-104, 1999.

CHOWDHURY, Gobinda G. **Natural language processing**. *Annual review of information science and technology* 37.1, p. 51-89, 2003.

DA SILVA, Cassiana F; VIEIRA , Renata. **Categorização de Textos da Língua Portuguesa com Árvores de Decisão, SVM e Informações Lingüísticas**. XXVII SBC, TIL-V Workshop em Tecnologia da Informação e da Linguagem Humana. 2007.

DAUMÉ III, Hal. A Course in Machine Learning. Disponível em: <http://ciml.info/dl/v0_9/>. Acesso em: 14 nov 2015.

DOMINGOS, Pedro. **A few useful things to know about machine learning.** Communications of the ACM, 55(10), p. 78–87, 2012.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. **From data mining to knowledge discovery in databases.** AI magazine 17.3, p. 37, 1996.

FELDMAN, Susan. **NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval.** ONLINE-WESTON THEN WILTON- 23, p. 62-73, 1999.

GABRILOVICH, Evgeniy; MARKOVITCH, Shaul. **Text categorization with many redundant features:** using aggressive feature selection to make SVMs competitive with C4. 5. Proceedings of the twenty-first international conference on Machine learning. ACM, 2004.

GANESAN, Kavita. Disponível em: <<https://bitbucket.org/kganes2/text-mining-resources/downloads>>. Acesso em: 14 jan 2016.

GOLDBERG, David E; HOLLAND, John H. **Genetic algorithms and machine learning.** *Machine learning* 3.2, p. 95-99, 1988.

HALL, Mark, et al. **The WEKA data mining software:** an update. ACM SIGKDD explorations newsletter 11.1, p. 10-18, 2009.

HARARI, Yuval N. **Sapiens: A brief history of humankind.** Random House. 2014.

HARPER, Douglas. On line etymology dictionary. Disponível em: <<http://www.etymonline.com/index.php?search=apprehend&searchmode=none>>. Acesso em: 14 de jan 2016.

HAUSER, Marc D; CHOMSKY, Noam; FITCH, William T. **The faculty of language: What is it, who has it, and how did it evolve?.** Science 298.5598, p. 1569-1579, 2002.

HOTHO, Andreas; NÜRNBERGER, Andreas; PAASS, Gerhard. **A Brief Survey of Text Mining.** Ldv Forum. Vol. 20, No. 1, 2005.

IKONOMAKIS, M; KOTSIANTIS, Sotiris; TAMPAKAS, V. **Text classification using machine learning techniques.** WSEAS Transactions on Computers 4.8, p. 966-974, 2005.

KOTSIANTIS, Sotiris B; ZAHARAKIS, I; PINTELAS, Panayiotis. **Supervised machine learning: A review of classification techniques.** p. 3-24, 2007

KRIESEL, David. **A brief introduction to neural networks.** 2007.

KYRIAKOPOULOU, Antonia; KALAMBOUKIS, Theodore. **Text classification using clustering.** Proceedings of the Discovery Challenge Workshop at ECML/PKDD, 2006.

LANGLEY, Pat. **Machine learning as an experimental science.** *Machine Learning* 3.1, p. 5-8, 1988.

LEWIS, David D. **Feature selection and feature extraction for text categorization.** Proceedings of the workshop on Speech and Natural Language. Association for Computational Linguistics, 1992.

LICHMAN, M. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. Disponível em: <<http://archive.ics.uci.edu/ml>>. Acesso em: 1 de maio de 2016.

LOVINS, Julie B. **Development of a stemming algorithm.** MIT Information Processing Group, Electronic Systems Laboratory, 1968.

MAHINOVS, Aigars, et al. **Text classification method review.** 2007.

MANNING, Christopher D; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **Introduction to information retrieval.** Online Edition. Cambridge: Cambridge University Press, 2009.

MARTIN, Dianne C. **ENIAC:** press conference that shook the world. *Technology and Society Magazine*, IEEE 14.4, p. 3-10, 1995.

MCCALLUM, Andrew; NIGAM, Kamal. **A comparison of event models for naive bayes text classification.** AAAI-98 workshop on learning for text categorization. Vol. 752. 1998.

MENGLER, Saket S. R; GOHARIAN, Nazli. **Detecting relationships among categories using text classification.** *Journal of the American Society for Information Science and Technology* 61.5, p. 1046-1061, 2010.

NIGAM, Kamal, et al. **Text classification from labeled and unlabeled documents using EM.** *Machine learning* 39.2-3, p. 103-134, 2000.

QUINLAN, J. Ross. **Induction of decision trees.** *Machine learning* 1.1, p. 81-106, 1986.

SHIER, Rosie. **Statistics: 1.1 Paired t-tests.** Disponível em: <<http://www.statstutor.ac.uk/resources/uploaded/paired-t-test.pdf>> Acesso em: 14 jun 2016.

SEBASTIANI, Fabrizio. **Text Categorization.** 2005.

SEGRE, Alberto M. **Applications of machine learning.** *IEEE Expert* 7.3, p. 31-34, 1992.

SKINNER, Burrhus F. **Are theories of learning necessary?.** *Psychological review* 57.4, p. 193, 1950.

SOARES, Fabio De Azevedo. **Mineração De Textos Na Coleta Inteligente De Dados Na Web.** Disponível em: <http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=13212@1>. Acesso em: 14 jan 2016.

WILBUR, W. John; SIROTKIN, Karl. **The automatic identification of stop words.** *Journal of information science* 18.1, p. 45-55, 1992.

APÊNDICE A – PROCESSO DE CLASSIFICAÇÃO DE TEXTOS COM WEKA SIMPLIFICADO

A seguir está descrito o processo de classificação a partir de arquivos de texto simples:

1) Obter os arquivos de texto e dividi-los dentro dos diretórios nomeados de acordo com as futuras classes desejadas.

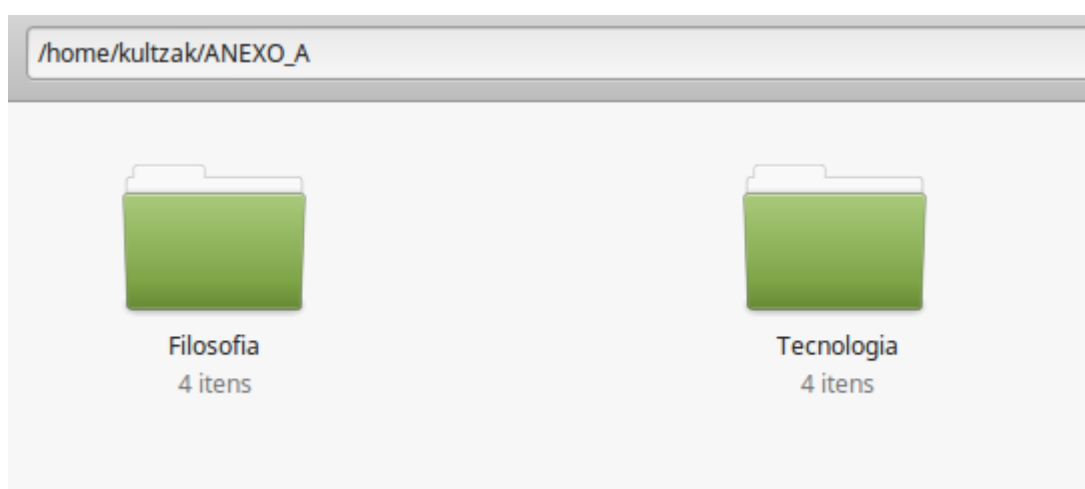


Figura 15 - Diretórios com arquivos de texto
Fonte: Autoria própria

Foram obtidos 4 textos com conteúdo relacionado a Filosofia e 4 relacionados a Tecnologia, de acordo com as Figuras 15 e 16.

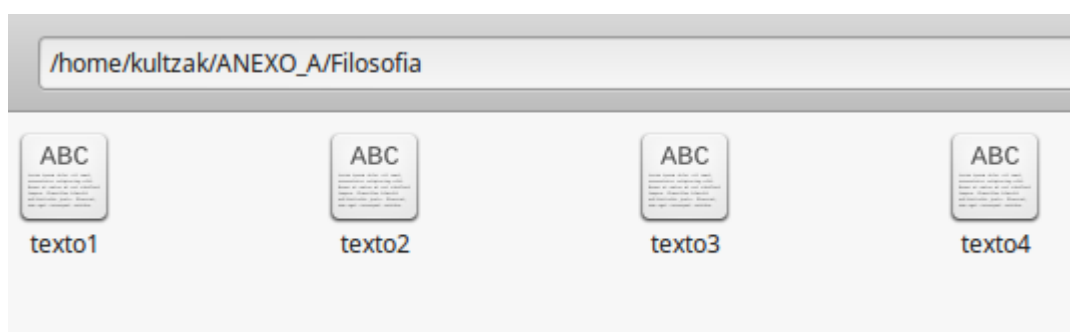
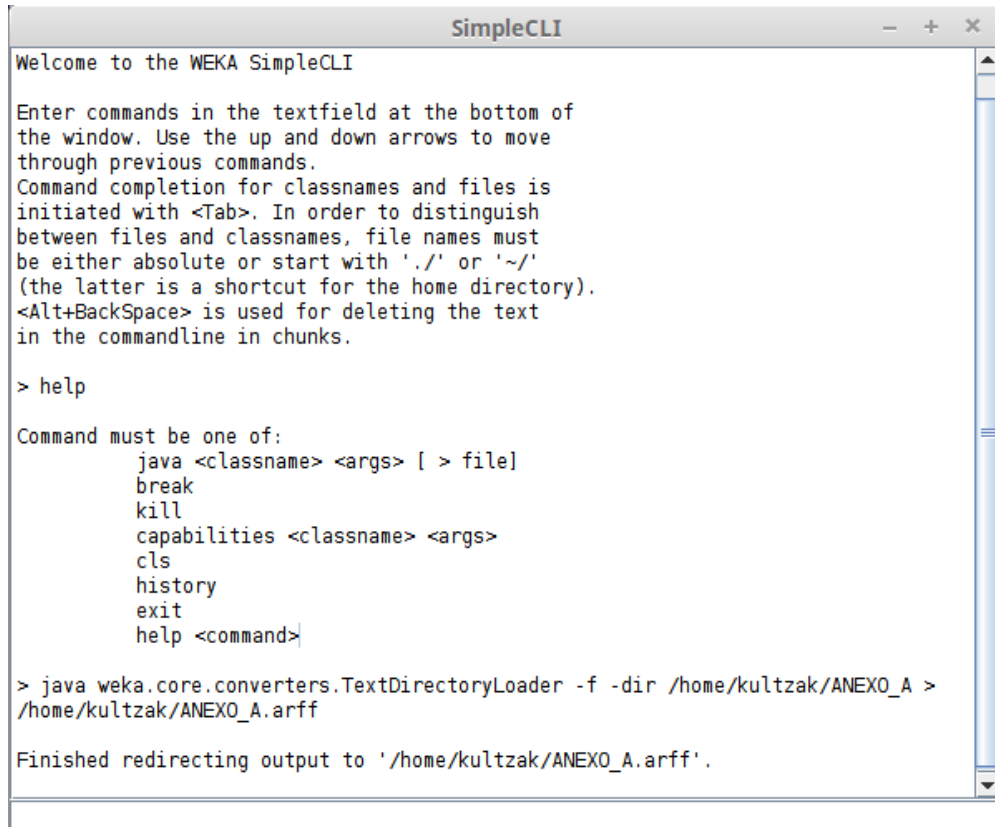


Figura 16 - Arquivos de texto
Fonte: Autoria própria

2) Aplicar o conversor *TextDirectoryLoader* da Interface *SimpleCLI* de acordo com o apresentado na Figura 17 será informando o diretório contendo os subdiretórios com os textos e o nome do arquivo de saída.



```

SimpleCLI
Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.
Command completion for classnames and files is
initiated with <Tab>. In order to distinguish
between files and classnames, file names must
be either absolute or start with './' or '~/
(the latter is a shortcut for the home directory).
<Alt+BackSpace> is used for deleting the text
in the commandline in chunks.

> help

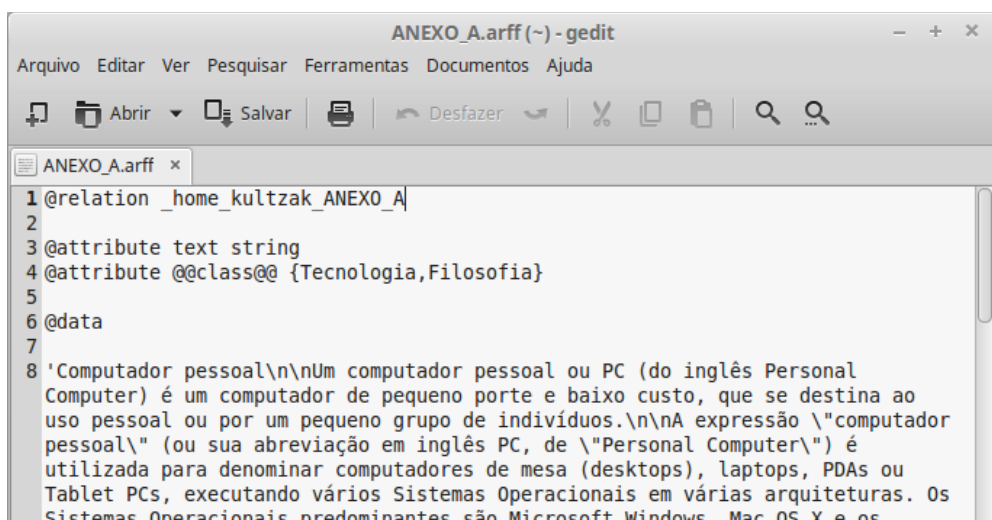
Command must be one of:
    java <classname> <args> [ > file]
    break
    kill
    capabilities <classname> <args>
    cls
    history
    exit
    help <command>

> java weka.core.converters.TextDirectoryLoader -f -dir /home/kultzak/ANEXO_A >
/home/kultzak/ANEXO_A.arff

Finished redirecting output to '/home/kultzak/ANEXO_A.arff'.
  
```

Figura 17 - Predições do algoritmo por instância
Fonte: Autoria própria

O Resultado será um arquivo .arff como o obtido da Figura 18.



```

ANEXO_A.arff (~) - gedit
Arquivo Editar Ver Pesquisar Ferramentas Documentos Ajuda
Abrir Salvar Desfazer
ANEXO_A.arff x
1 @relation _home_kultzak_ANEXO_A
2
3 @attribute text string
4 @attribute @@class@@ {Tecnologia,Filosofia}
5
6 @data
7
8 'Computador pessoal\n\nUm computador pessoal ou PC (do inglês Personal
Computer) é um computador de pequeno porte e baixo custo, que se destina ao
uso pessoal ou por um pequeno grupo de indivíduos.\n\nA expressão "computador
pessoal" (ou sua abreviação em inglês PC, de "Personal Computer") é
utilizada para denominar computadores de mesa (desktops), laptops, PDAs ou
Tablet PCs, executando vários Sistemas Operacionais em várias arquiteturas. Os
Sistemas Operacionais predominantes são Microsoft Windows, Mac OS X e os
  
```

Figura 18 - Arquivo .arff gerado
Fonte: Autoria própria

3) Através da interface *Explorer* abrir o arquivo *.arff* obtido na etapa anterior, como consta na Figura 19.

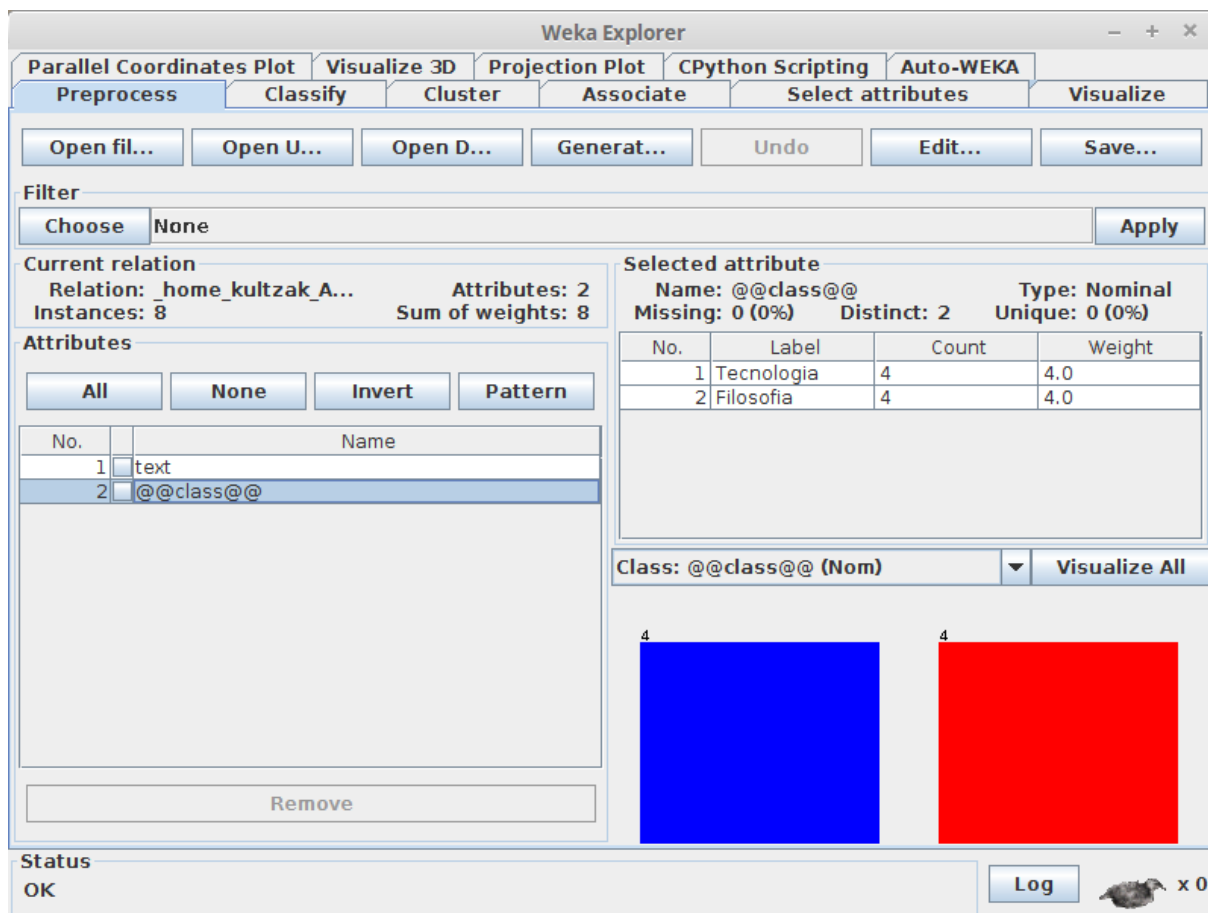


Figura 19 - Abertura do arquivo *.arff* através do WEKA Explorer
Fonte: Autoria própria

Pode-se verificar a presença das duas classes (Filosofia, Tecnologia) com 4 instancias cada, porém os atributos ainda não foram extraídos dos textos.

4) Selecionar o filtro através do caminho Filters > unsupervised > attribute > StringToWordVector.

Aqui podem ser selecionadas configurações referentes à normalização das classes, aplicação de lista de *stopwords*, algoritmo de *stemming*, entre outras opções. neste exemplo foram utilizadas as configurações *default* de acordo com a Figura 20.

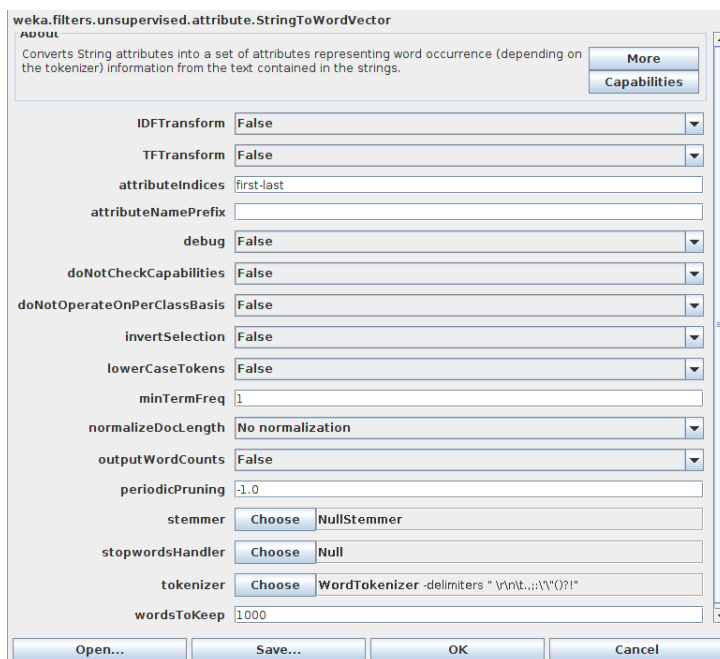


Figura 20 - Tela StringToWordVector
Fonte: Autoria própria

5) Depois de aplicado o filtro *StringToWordVector* o atributo correspondente a classe fica na primeira posição, o que durante a classificação retorna um erro, deve-se selecionar a opção *Edit*, clicar com o botão direito sobre a coluna correspondente à classe e selecionar a opção *Attribute as class*.

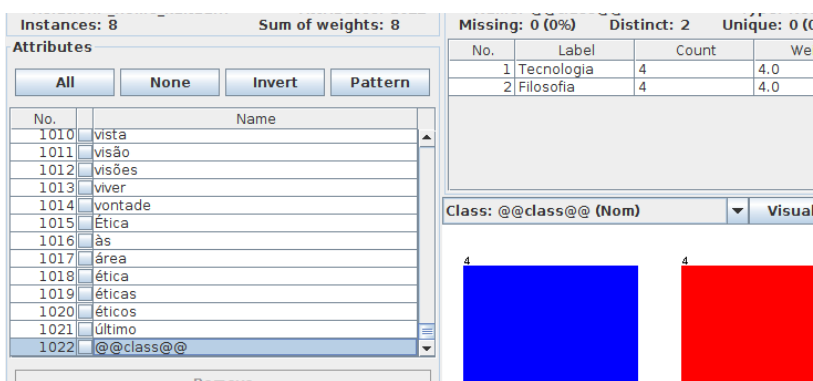


Figura 21 - Último atributo corresponde às classes
Fonte: Autoria própria

Observa-se que o atributo referente a classe assume a última posição, conforme a figura 21.

6) A base de dados está pronta para aplicação do algoritmo de classificação.

- Selecionar a aba *Classify* e escolher o algoritmo de classificação desejado;
- Informar as opções para o teste da base de dados;
- Para a visualização dos resultados previstos em cada documento é necessário selecionar através de *More options > Output predictions* e informar a forma de saída desejada;
- Apertar *Start* para iniciar.

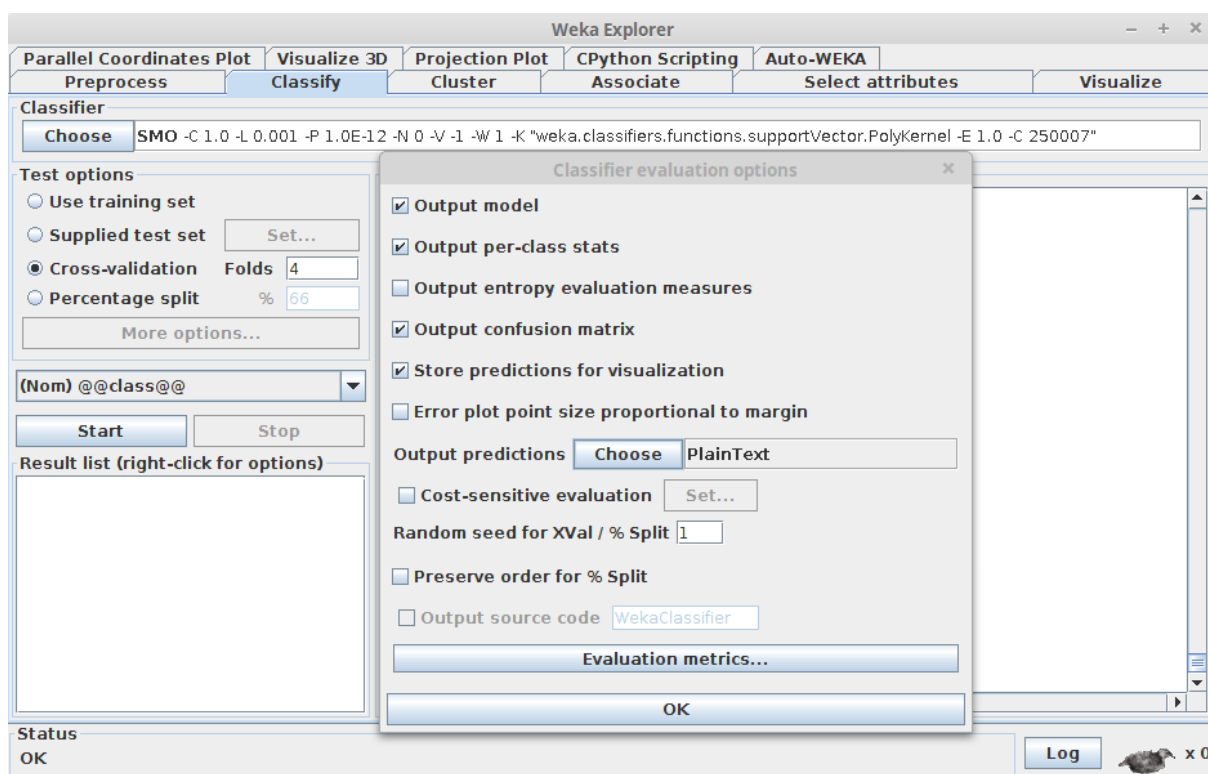


Figura 22 - Opções para classificação
Fonte: Autoria própria

No caso da Figura 22 foi selecionado o método de *Cross-validation* com 4 *folds*, tendo em vista que o número de *folds* não pode ser maior do que o número de instâncias.

A Figura 23 apresenta a quantidade de documentos classificados corretamente e na sequência os resultados de diversos tipos de avaliação de eficiência para cada classe.

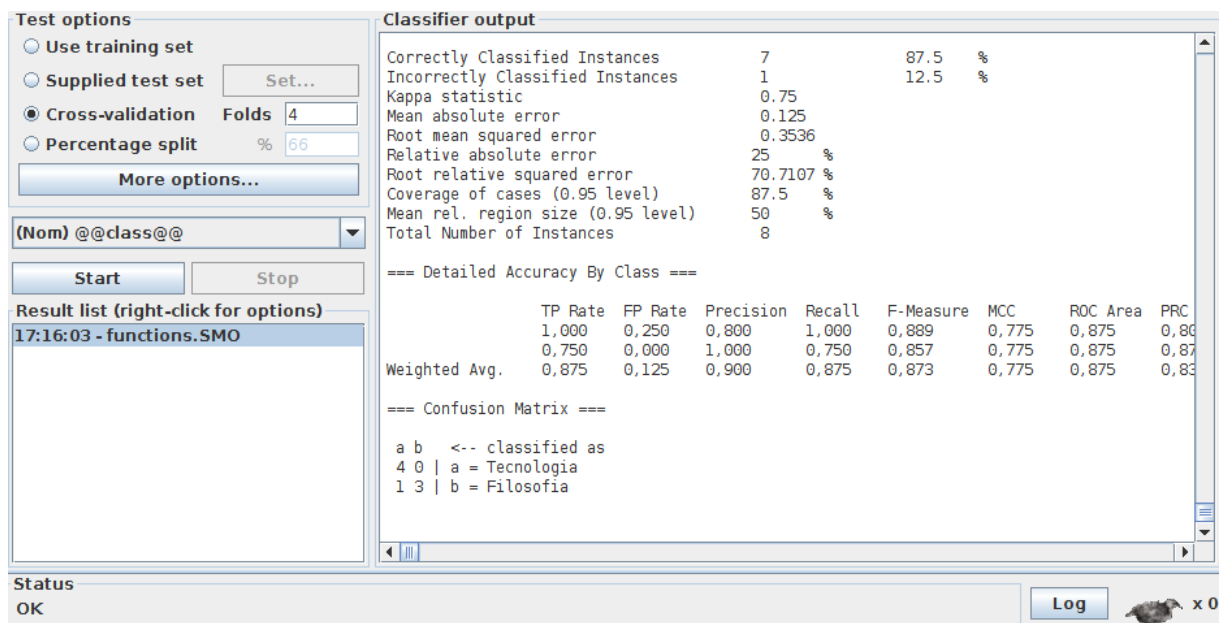


Figura 23 - Resultado da classificação
 Fonte: Autoria própria

Ao analisar os resultados observa-se que 7 das 8 instâncias foram classificadas corretamente.

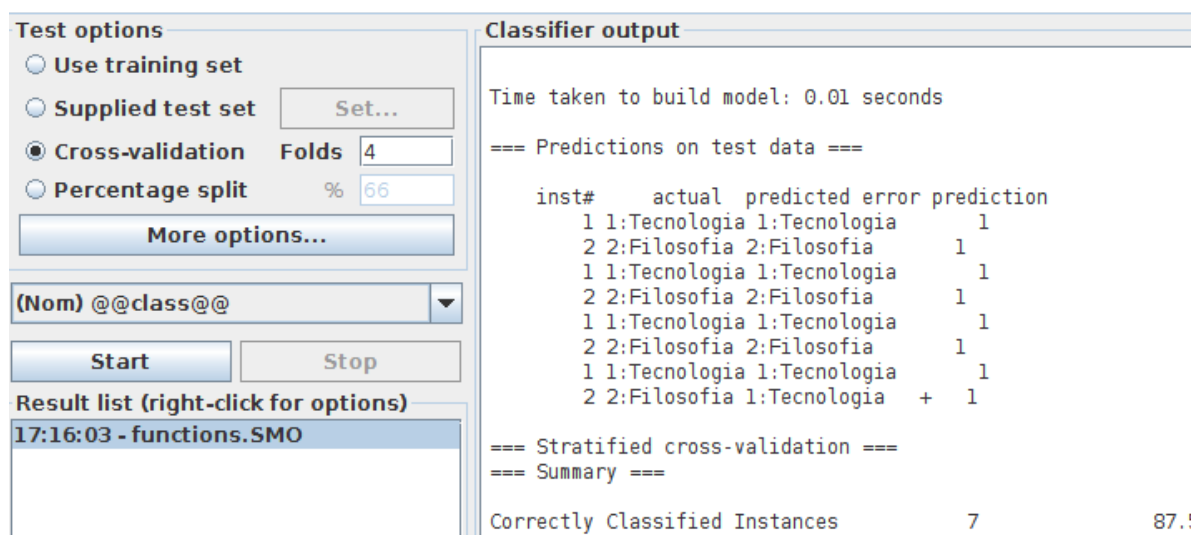


Figura 24 - Predições do algoritmo selecionado
 Fonte: Autoria própria

A Figura 24 mostra que a última instância a ser classificada como Filosofia foi incorretamente atribuída a classe Tecnologia.