UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MURILO GATTO

RECOMENDAÇÃO AUTOMÁTICA DE TEXTOS PARA CRIANÇAS ENTRE A TERCEIRA E SÉTIMA SÉRIE DO ENSINO FUNDAMENTAL

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2016

MURILO GATTO

RECOMENDAÇÃO AUTOMÁTICA DE TEXTOS PARA CRIANÇAS ENTRE A TERCEIRA E SÉTIMA SÉRIE DO ENSINO FUNDAMENTAL

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Arnaldo Candido Júnior

MEDIANEIRA

2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

RECOMENDAÇÃO AUTOMÁTICA DE TEXTOS PARA CRIANÇAS ENTRE A TERCEIRA E SÉTIMA SÉRIE DO ENSINO FUNDAMENTAL

Por

Murilo Gatto

Este Trabalho de Diplomação (TD) foi apresentado às 13:00 h do dia 18 de novembro de 2016 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

Prof. Dr. Arnaldo Candido Júnior
UTFPR – *Campus* Medianeira
(Orientador)

Prof. Me. Jorge Aikes Junior
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Me. César Angonese
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Me. Jorge Aikes Junior
UTFPR – *Campus* Medianeira
(Responsável pelas atividades de TCC)

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

GATTO, Murilo. RECOMENDAÇÃO AUTOMÁTICA DE TEXTOS PARA CRIANÇAS ENTRE A TERCEIRA E SÉTIMA SÉRIE DO ENSINO FUNDAMENTAL Trabalho de Diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas). Universidade Tecnológica Federal do Paraná. Medianeira 2016.

Motivar os alunos entre a terceira e sétima série do ensino fundamental a continuar na escola, é algo complexo. Um dos motivos é a dificuldade de encontrar um material didático apropriado para essa fase do aprendizado. Devido a isso, é proposto uma forma de recomendar automaticamente textos para esses novos leitores, utilizando técnicas de mineração de textos juntamente com Aprendizado de Máquina. Serão descritas e utilizadas várias técnicas a fim de validar a confiabilidade dessa metodologia, a qual espera ajudar na redução do abandono escolar. Até o momento alguns experimentos preliminares já foram realizados.

Palavras-Chave: Classificação Automática de Textos. Recomendação de Conteúdo. Aprendizado de Máquina.

ABSTRACT

GATTO, Murilo. RECOMENDAÇÃO AUTOMÁTICA DE TEXTOS PARA CRIANÇAS ENTRE A TERCEIRA E SÉTIMA SÉRIE DO ENSINO FUNDAMENTAL Trabalho de Diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas). Universidade Tecnológica Federal do Paraná. Medianeira 2016.

In Brazil, it is complex to Motivate students between the third and seventh grade of primary school to continue in school. One reason is the difficulty in finding appropriate didactic material for this learning phase. Because of this, this work proposes a way to recommend automatically texts to these new readers using texts mining techniques together with Machine Learning approaches. Various techniques will be described and used in order to validate the reliability of this method, which is expected to help in reducing school dropout. Until now some preliminary experiments have been performed.

Palavras-Chave: Automatic classification of texts. Content recommendation. Machine Learning.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

AM	Aprendizado de Máquina
ANSI	*American National Standards Institute*
FUNTEF	Fundação de Apoio à Educação, Pesquisa e Desenvolvimento Cientifico e Tecnológico da Universidade Tecnológica Federal do Paraná
HTML	*HyperText Markup Language*
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	Integrated Development Environment
MLP	Multilayer Perceptron
NILC	Núcleo Interinstitucional de Linguística Computacional
PHP	PHP: Hypertext Preprocessor
PLN	Processamento de Linguagem Natural
RNA	Rede Neural Artificial
SVM	Support Vector Machine
USP	Universidade de São Paulo
UTF-8	8-bit *Unicode Transformation Format*
WEKA	Waikato Environment fo Knowledge Analysis

SUMÁRIO

1 INTRODUÇÃO

Crianças em fase de alfabetização têm dificuldades na leitura da maior parte dos textos disponíveis, já que eles foram concebidos inicialmente para leitores com capacidade plena de leitura. Fornecer textos adequados para tal público alvo ajuda a motivar e estimular esses novos leitores durante sua educação.

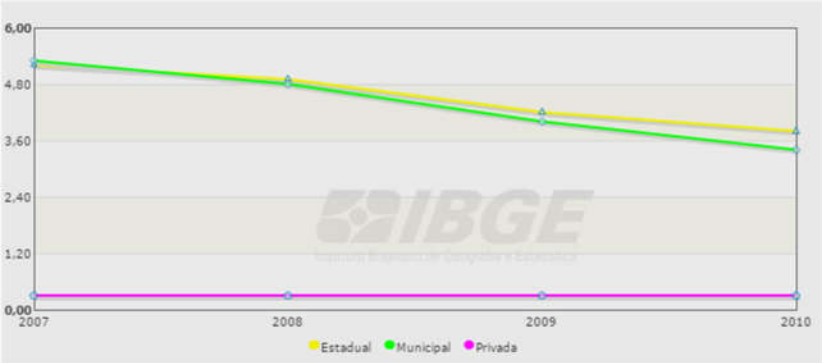

Figura 1 - Abandono Escolar por dependência administrativa

Fonte: IBGE (2016)

No Brasil a tarefa é particularmente importante pois, dados do IBGE (2016) (Instituto Brasileiro de Geografia e Estatística) indicam que o índice de abandono escolar é alto, e por mais que tenha sido reduzido nos últimos anos, conforme ilustra a Figura 1, ficou em 3,1% no ano de 2010, tendo índices mais altos na região Norte e Nordeste, chegando a 8,7% no estado do Alagoas, conforme ilustrado na Figura 2. Além disso estatísticas comprovam que a maior parte do abandono escolar é oriundo de escolas estaduais e municipais, sendo, respectivamente, 3,8% e 3,4% contra 0,3% da escola privada.

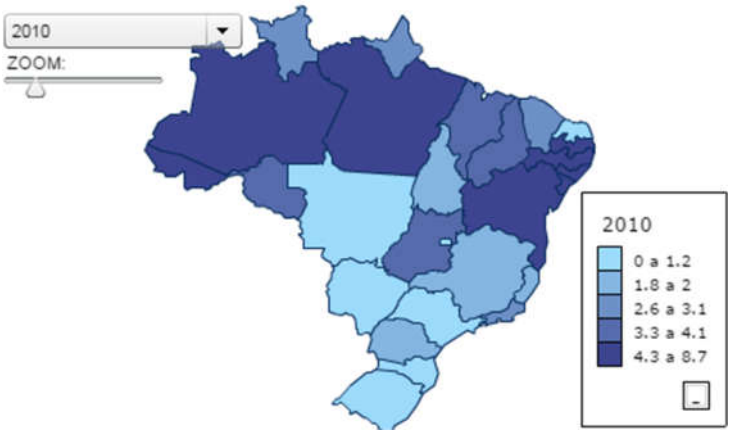

Figura 2 - Abandono escolar por estado

Fonte: IBGE (2016)

Dados dessa mesma fonte também indicam que o índice mais alto de abandono é na quinta série (sexto ano) do ensino fundamental, em média 5,1% chegando a ultrapassar os 10% em alguns estados conforme ilustra a Figura 3.

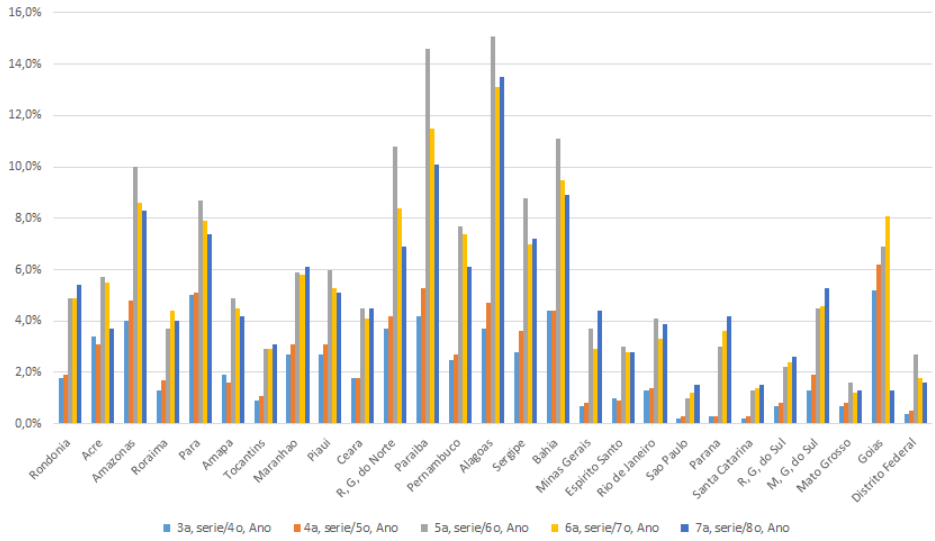

Figura 3 - Abandono escolar por estado x série

Fonte: Adaptado de (IBGE, 2016)

Uma estratégia para desestimular o abandono é tentar motivar os alunos na leitura através de textos pré-selecionados para esse público alvo. Este trabalho tem como finalidade classificar textos automaticamente para alfabetizandos entre a terceira e a sétima série do ensino fundamental. Para isso, pretende-se aplicar técnicas de Aprendizado de Máquina (AM) em um problema do Processamento de Linguagem Natural (PLN).

Este trabalho foi originalmente proposto para prover subsídios ao projeto 22/2015 da FUNTEF (Fundação de Apoio à Educação, Pesquisa e Desenvolvimento Científico e Tecnológico da Universidade Tecnológica Federal do Paraná) **personalização da leitura por meio de ferramentas de classificação automática da complexidade e adaptação textual**, que por sua vez, está integrado em uma parceria com a empresa Guten Tecnologia e a USP (Universidade de São Paulo). Atualmente, a referida empresa já conta com um classificador para a tarefa, cuja acurácia é de 47%. Neste trabalho, investiga-se atributos adicionais para melhorar o desempenho do referido classificador.

1.1 OBJETIVO GERAL

Utilizar técnicas de pré-processamento de textos e aprendizado de máquina para classificar o nível de dificuldade de textos tendo em vista leitores da terceira à sétima série.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um software para gerar uma *bag-of-words* e efetuar o corte de Luhn;
- Utilizar técnicas de aprendizado de máquina, em particular, classificação para classificar automaticamente os textos;
- Avaliar criticamente a capacidade do classificador obtido;

1.3 JUSTIFICATIVA

Este trabalho visa ajudar na redução do abandono escolar trabalhando sobre um de seus inúmeros motivadores. Para isso é trabalhado sobre a hipótese de que o método utilizado atualmente nas escolas para a seleção do material para exercitar a leitura de crianças em fase de alfabetização pode estar sendo ineficiente.

Recomendar textos adequados para crianças entre a terceira e a sétima série pode ser um grande desafio, pois além de escolher textos que essas crianças em fase de alfabetização compreendam, é necessário escolher textos que os motivem a continuar lendo e estudando. Além disso, existe um imenso volume de textos disponíveis, o que torna ainda mais complexo recomendar textos adequados para esse público alvo. Esse trabalho tentará ajudar nesse processo. Um classificador de textos foi desenvolvido e treinado, automatizando assim essa tarefa.

Conforme os dados citados anteriormente, a evasão escolar chega a 8.7% em alguns pontos do Brasil, baseando-se nessa informação espera-se que no final desse projeto tenha-se uma metodologia capaz de recomendar textos adequadamente para essas crianças, o que pode, mesmo que pouco, a ajudar motivar esses pequenos leitores a continuarem estudando.

1.4 ORGANIZAÇÃO DO TEXTO

Esse trabalho é organizado como segue. O Capítulo 2 detalha o referencial teórico do projeto, abrangendo a área de aprendizado de máquina. O Capítulo 3 apresenta uma visão geral dos materiais e métodos utilizados. Já no Capítulo 4 são relatados os experimentos, e seus resultados são apresentados e discutido e por fim o Capítulo 5 detalha as considerações finais.

2 REVISAO BIBLIOGRÁFICA

Antes da década de 1970, as pesquisas sobre Inteligência Artificial (IA) eram apenas teóricas, sendo aplicada somente em pequenos problemas curiosos, com pouco valor prático. Porém na década de 1970 houve uma maior disseminação e uso dessas técnicas (FACELI et al., 2011).

O uso da IA está muito presente no dia a dia das pessoas, por exemplo, em redes sociais esses algoritmos analisam o perfil de acesso dos usuários para lhes fornecer um conteúdo mais adequando com seus perfis. Outro uso bem disseminado é o marketing digital, onde a IA exibe ofertas de alguns produtos com base no que o usuário já visualizou previamente. Esse processo de adquirir experiência e induzir uma hipótese é chamado de AM (Aprendizado de Máquina).

Segundo Arthur Samuel (1959) o aprendizado de máquina pode ser definido como uma área de estudo que fornece aos computadores a habilidade de aprender sem serem programados. Já Mitchell (1997) dá a seguinte definição operacional:

"Um programa de computador aprende com experiência *E* com respeito a tarefa *T* e medida de desempenho *P*, se a performance na tarefa *T*, mensurada por *P*, melhora com experiência *E*."

Mitchell (1997) ainda traz alguns exemplos bem-sucedidos do uso do aprendizado de máquina.

- Sistemas de reconhecimento de fala;
- Carros Autônomos;
- Classificadores de estruturas astronômicas.;
- Jogar jogos de tabuleiro em torneios contra especialistas;

Nas seguintes seções serão abordados alguns métodos que utilizam os princípios da IA e do AM.

2.1 REDES NEURAIS ARTIFICIAIS

Quando comparados aos computadores, o cérebro humano apresenta muitas características interessantes, entre elas o paralelismo massivo para reconhecer padrões, capacidade de aprendizado e generalização, adaptabilidade,

tolerância a falhas, processamento distribuído, baixo consumo de energia. As Redes Neurais Artificiais (RNAs) tiveram como inspiração o cérebro humano, particularmente inspiradas nas características apresentadas (ROSA, 2011).

Os dados presentes na Tabela 1 ilustram a diferença entre os recursos computacionais disponíveis em um supercomputador, utilizado especificamente o Blue Gene da IBM nesse exemplo, um computador pessoal do ano de 2008, e o cérebro humano.

	Supercomputador	Computador Pessoal	Mente Humana
Unidades Computacionais	10^4 CPUs, 10^{12} transistores	4 CPUs, 10^9 transistores	10^{11} neurônios
Unidades de armazenamento	10^{14} bits RAM	10^{11} bits	10^{11} neurônios
	10^{15} bits disco	10^{13} RAM bits disco	10^{14} sinapses
Tempo de Ciclo	10^{-9} seg	10^{-9} seg	10^{-3} seg
Operações/seg	10^{15}	10^{10}	10^{17}
Atualizações de memória/seg	10^{14}	10^{10}	10^{14}

Tabela 1 - Comparação de recursos computacionais

Fonte: Adaptado de (RUSSEL, 2013)

Esses valores, para um supercomputador, tendem a crescer por um fator de 10 a cada 5 anos aproximadamente. Já na mente humana, esses valores têm se mantido essencialmente estáveis. O computador pessoal apenas vence o cérebro no quesito tempo de ciclo (RUSSEL, 2013).

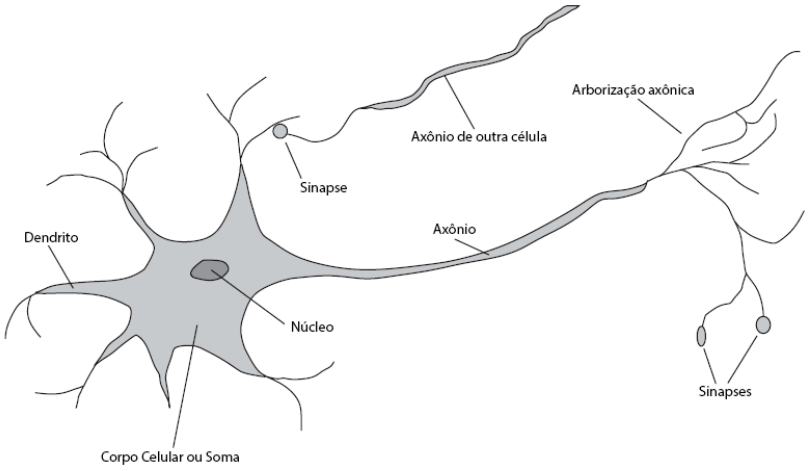

Figura 4 - Neurônio Biológico

Fonte: Adaptado de Russel (2013)

A Figura 4 traz um exemplo simplificado de neurônio biológico, cujo seus principais componentes são dendritos, corpo celular e axônio. Os dendritos são

responsáveis por receber informações de outros neurônios ou de células sensoras. O axônio permite enviar dados para outros neurônios, bem como para sistemas motores do organismo. À comunicação entre axônio e dendrito dá-se o nome de sinapse. O corpo celular faz o processamento dos dados recebidos, e dependendo do resultado comunica com o próximo neurônio da cadeia.

O cérebro humano possui um número massivo de neurônios, da ordem de 10 bilhões a 500 bilhões, e cada um desses neurônios pode estar ligado a centenas ou até milhares de outros. Devido a isso os neurônios funcionam de uma forma massivamente paralela, provendo uma grande rapidez de processamento, e possibilitando realizar diversas tarefas muito mais rápido do que o computador digital (FACELI et al., 2011).

O neurônio artificial simula o neurônio biológico, sendo a unidade de processamento principal de uma RNA (Rede Neural Artificial). Na Figura 5, é ilustrado um esquema de um neurônio artificial, no qual x_m são os valores de entrada recebidos por ele, w_m são os pesos sinápticos atribuídos a esses valores, b_k é o bias, ou seja, um valor fixo assumido no processamento, tendo o efeito de aumentar ou diminuir a entrada da função de ativação. ∑ é um somador, o qual tem a função de combinar os sinais de entrada ponderando-os por seus respectivos pesos, e a Função de ativação (φ) serve para restringir a saída do neurônio.

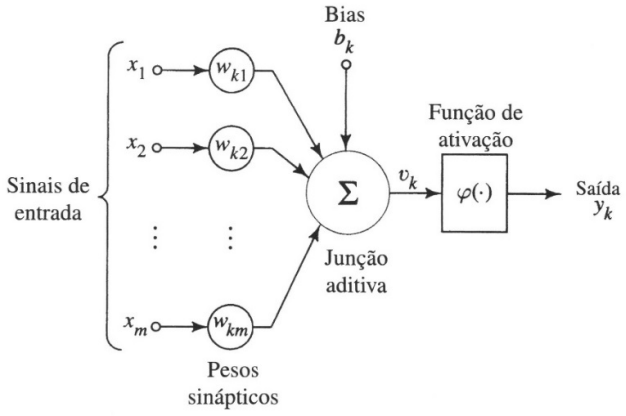

Figura 5 - Neurônio Artificial

Fonte: Haykin (2000)

Esse mesmo neurônio pode ser descrito matematicamente conforme as equações (1) e (2). Os atributos são multiplicados pelos seus pesos resultando no valor u_k. O valor u_k, por sua vez, é somado ao bias e apresentado à função de ativação.

$$v_k = \sum_{j=}^{m} w_{kj} x_j \quad \textbf{(1)}$$

$$y_k = \varphi(u_k + b_k) \quad \textbf{(2)}$$

Em uma RNA é possível a existência de mais de uma camada de neurônios, os quais ligam seus terminais de saída nos de entrada dos neurônios da camada seguinte. Esse tipo de RNA é chamada de rede multicamada, a qual é composta por uma ou mais camadas intermediárias e uma camada de saída (FACELI et al., 2011).

As próximas subseções detalham RNAs: Perceptron, MLP (*MultiLayer Perceptron* – Perceptron Multicamada)

2.1.1 *Perceptron*

Utilizando o modelo de McCulloch-Pitts como neurônio e desenvolvida por Rosenblatt em 1958, a rede Perceptron foi a primeira RNA a ser implementada. Embora ela seja composta por apenas uma camada de neurônios, apresenta um bom resultado em diversos problemas de classificação de padrões linearmente separáveis (ROSA, 2011). Na Figura 6(a) é ilustrado um conjunto de dados linearmente separável (OU-Lógico), e, na Figura 6(b) é ilustrado um conjunto de dados que não pode ser separado linearmente (OU-Exclusivo), sendo os pontos brancos pertencente a uma classe e os pretos a outra.

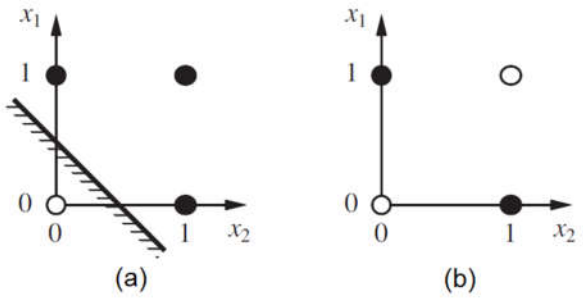

Figura 6 – Conjunto linearmente separável (a) e conjunto não separável linearmente (b)

Fonte: Adaptado de Russel (2013)

Segundo Faceli et al. (2011), alguns anos após essa rede ser proposta, Rosenblatt provou o teorema de convergência da rede Perceptron, o qual diz: “Se é

possível classificar um conjunto de entradas linearmente, uma rede Perceptron fará a classificação".

Um Perceptron pode ser utilizado para realizar tarefas, por exemplo, classificação de imagens e alguns reconhecimentos simples. Devido a isso ele pode ter inúmeras entradas, que em alguns casos podem ser organizados em grade, a qual pode representar um campo de visão ou uma imagem (COPPIN, 2015).

2.1.2 *Perceptron Multicamada*

A rede Perceptron é muito boa para resolver problemas linearmente separáveis, mas por ser composta de apenas uma camada, não tem tanta acurácia em situações que não podem ser separadas linearmente. Segundo Cybenko (1989 apud FACELI et al., 2011, p. 115): "uma rede com uma camada intermediária pode implementar qualquer função contínua. A utilização de duas camadas intermediárias permite a aproximação de qualquer função".

função APRENDIZAGEM-DE-RETRO-PROP(*exemplos*, *rede*) **retorna** uma rede neural
entradas: *exemplos*, um conjunto de exemplos, cada um com vetor de entrada **x** e vetor de saída **y**
rede, uma rede multicamadas com L camadas, pesos $w_{i,j}$ e unção de ativação g
variáveis locais: Δ, um vetor de erros, indexado pelo nó de rede

repetir
  para cada peso $w_{i,j}$ na *rede* **faça**
    $w_{i,j} \leftarrow$ um número randômico pequeno
  para cada exemplo (**x**, **y**) **em** *exemplos* **faça**
    /* *Propagar as entradas para a frente para computar as saídas* */
    para cada nó i na camada de entrada **faça**
      $a_i \leftarrow x_i$
    para $\ell = 2$ **até** L **faça**
    para cada nó j na camada ℓ **faça**
      $in_j \leftarrow \sum_i w_{i,j}\, a_i$
      $a_j \leftarrow g(in_j)$
    /* *Propagar deltas retrocedendo da camada de saída para a camada de entrada* */
    para cada nó j na camada de saída **faça**
      $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
    para $\ell = L - 1$ **até** 1 **faça**
      para cada nó i na camada ℓ **faça**
        $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j}\, \Delta[j]$
    /* *Atualiza cada peso na rede usando deltas* */
    para cada peso wi,j na rede **faça**
      $w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
até que algum critério de parada seja satisfeito
retorno *rede*

Código 1 - Algoritmo Backpropagation

Fonte: Russel (2013)

O grande desafio para que RNAs com múltiplas camadas fossem desenvolvidas era a capacidade de treinamento, desafio esse que foi transposto pelo algoritmo de retropropagação de erros (*error backpropagation*), o qual é ilustrado pelo Código 1. Esse algoritmo de treinamento funciona em duas etapas, primeiramente a etapa *forward* (para frente), na qual os dados de entrada são apresentados à primeira camada da rede, após isso suas saídas são apresentadas como entrada da camada seguinte. Esse processo se repete até que a camada de saída produza o resultado final do ciclo, o qual é comparado com o valor desejado. A diferença entre esses valores indica o erro cometido pela RNA, erro esse que é utilizado na fase *backward* (para trás) para ajustar os pesos dos neurônios (FACELI et al., 2011).

2.2 MÁQUINAS DE VETORES DE SUPORTE

As Maquinas de Vetores de Suporte, ou SVM (*Support Vector Machuine*) ganharam nos últimos tempos um destaque na comunidade de AM, pois seus resultados são comparáveis e muitas vezes superiores aos obtidos utilizando outras técnicas, como por exemplo RNAs (FACELI et al., 2011).

Na Figura 7 é ilustrado a diferença entre o processamento de uma RNA (a) e uma SVM (b), ambas aplicadas a duas classes. Os pontos pretos representam uma classe e os pontos brancos outra. Analisando a imagem percebe-se que a RNA pode traçar várias linhas de separação, satisfazendo-se com qualquer uma delas, porém em alguns casos a linha é traçada muito perto do conjunto de dados, deixando uma margem muito baixa para variação. Já a SVM trabalha baseada na teoria de aprendizado estatístico, o qual utiliza um separador de margem máxima. A margem é a área (ou hiperplano), formado entre as duas linhas pontilhadas, as quais são chamadas de vetores de suporte (RUSSEL, 2013).

As SVM podem ser classificadas de duas formas:

- SVM com margens Rígidas: utilizada em dados linearmente separáveis, assim restringindo que hajam no hiperplano instancias entre os vetores de suporte;

- SVM com margens suaves: utilizada quando os dados não são linearmente separáveis, permitindo que ruídos na fronteira de decisão sejam ignorados;

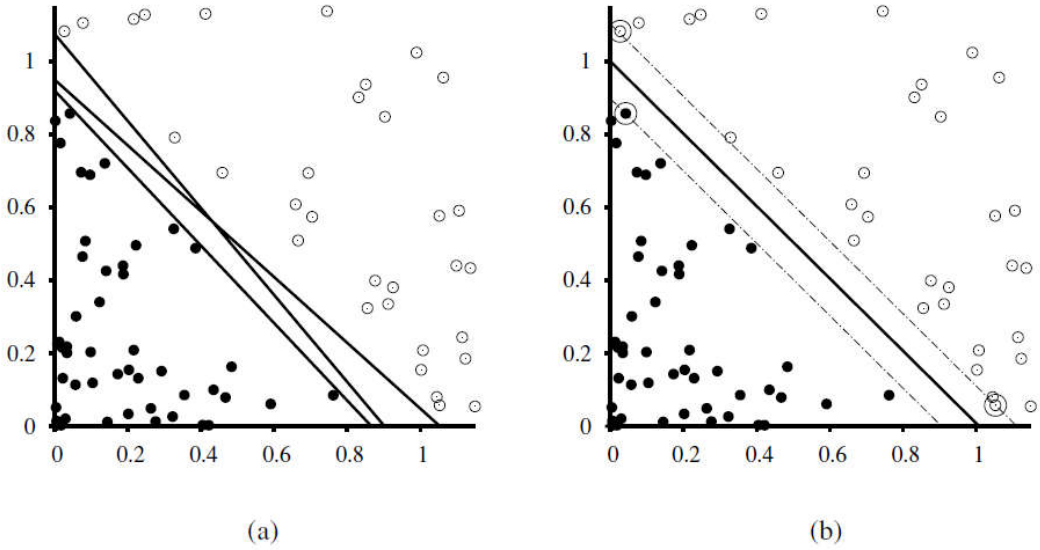

Figura 7 - Forma de classificação utilizada por uma RNA (a) e por uma SVM(b)

Fonte: Russel (2013)

A Figura 8 ilustra a diferença entre esses dois tipos de SVMs.

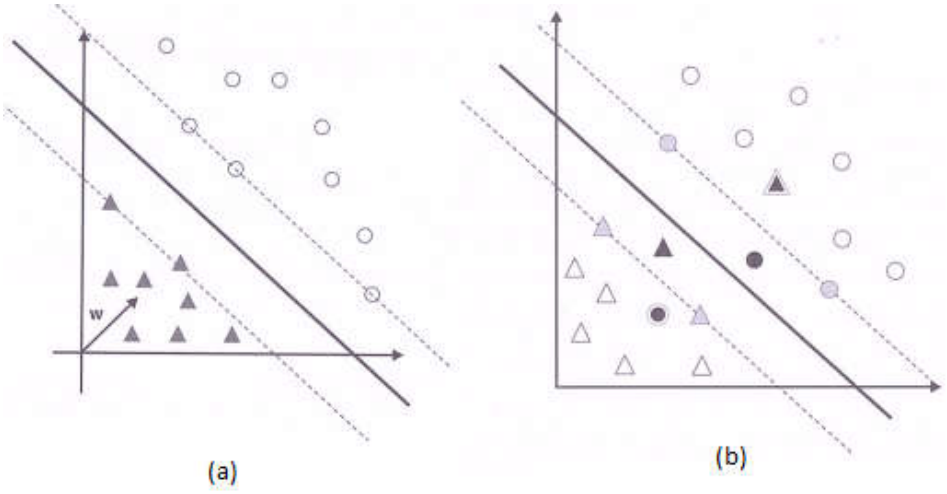

Figura 8 - SVM de margem rígida (a) e SVM de margem suave (b)

Fonte: Adaptado de Faceli et al. (2011)

Segundo Russel (2013), quando os dados a serem classificados não são linearmente separáveis, as SVMs utilizam uma função kernel, na qual a dimensão dos

dados é alterada, assim se tornando linearmente separáveis. A Figura 9 ilustra essa situação, onde em duas dimensões os dados não são linearmente separáveis, porém, em três dimensões eles são.

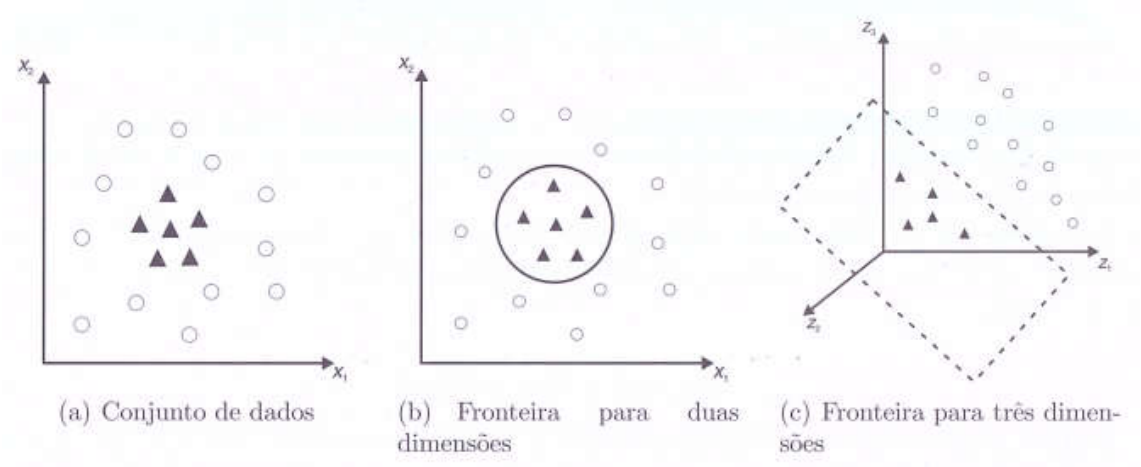

Figura 9 - Fronteiras de dados em dimensões diferente

Fonte: Faceli et al. (2011)

Segundo Herbrich (2001 apud FACELI et al., 2011, p. 132) a função kernel pode ser definida da seguinte forma: “é uma função que recebe dois pontos x_i e x_j no espaço de entradas e calcula o produto escalar desses objetos no espaço de características”.

2.3 PRÉ-PROCESSAMENTO DE TEXTOS

Uma das complexidades para o Processamento de Língua Natural (PLN), é a grande dimensão dos conjuntos de dados, dessa forma, um dos objetivos do pré-processamento de textos é reduzir o número de termos (ARANHA, 2007). Algumas das técnicas que podem ser utilizadas serão relatadas nas seguintes subseções.

2.3.1 Remoção de *stop-words*

Um dos recursos utilizados para a redução do número de termos do conjunto de dados durante o pré-processamento de textos é a remoção das *stop-words*, pois essas palavras normalmente aparecem em muitos documentos e podem comprometer a precisão e a eficiência dos sistemas de IA (ARANHA, 2007).

Um exemplo desse tipo de palavra são as preposições. Esses termos são utilizados para fazer o encadeamento de ideias e palavras, e segundo Aranha (2007), eles não são inerentes ao conteúdo dos textos, e sim a sua linguagem.

Essas palavras podem ser obtidas de forma manual, onde o desenvolvedor seleciona as palavras que não devem ser incluídas no dicionário, ou automática. Nos experimentos realizados foi utilizada uma lista de *stop-words* encontrada no site da empresa Intext[1].

2.3.2 Corte de Luhn

Segundo Aranha (2007), uma estratégia muito citada na literatura é a utilização da Lei de Zipf e corte de Luhn.

A lei de Zipf é uma dei de potências proposta por George Kingsley Zipf, que era linguista na universidade de Harvard e viveu entre 1902 e 1950. Ela propõe que a frequência de ocorrência de algum evento está relacionada a uma função de ordenação (MARTINS; MONARD; MATSUBARA, 2003).

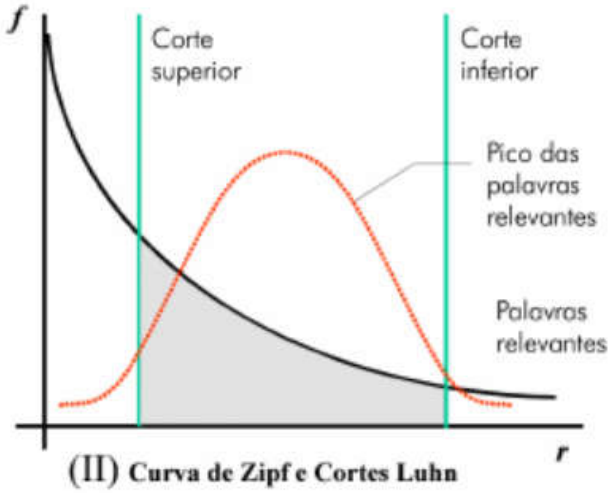

Figura 10 - Curva de Zipf e Cortes de Luhn

Fonte: Aranha (2007)

A lei de Zipf demonstra que vários fenômenos seguem uma distribuição similar, sendo a linguagem um desses fenômenos. Para uma coleção de textos, uma

[1] http://www.intext.com.br/

das formas de se enunciar a lei de Zipf é contar o número de aparições de cada palavra, e ordená-las de forma crescente.

Baseando-se na lei de Zipf, Hans Peter Luhn, propôs dois cortes. No corte superior são removidas as palavras mais frequentes, pois são encontradas na maioria dos textos, já no corte inferior são removidas as palavras menos frequentes, que podem ser consideradas raras, não contribuindo significantemente para a discriminação dos textos. No centro dos dois cortes, são encontrados os termos mais relevantes. A Figura 10 ilustra a curva de Zipf, e os cortes de Luhn (MARTINS; MONARD; MATSUBARA, 2003).

3 MATERIAIS E MÉTODOS

Nesse capítulo serão relatados os materiais que foram utilizados para a realização dos experimentos, além da metodologia utilizada para a execução dos mesmos.

3.1 METODOLOGIA

Vale destacar que para não haver discrepâncias nos resultados, após ter o arquivo com as informações a serem processadas pelos classificadores, foi adotada o seguinte método:

- Primeiramente os dados passaram pelo classificador SMO;
- Após o processamento do classificador SMO foi utilizado o classificador MLP;
- Utilizado o recurso de Seleção de Atributos, com o algoritmo CfsSubsetEval;
- Os dados gerados pela Seleção de Atributos foram submetidos novamente ao classificador SMO;
- Por fim, os mesmos dados foram submetidos novamente ao classificador MLP;

Outro ponto que é valido destacar é que a versão do WEKA utilizado é a 3.8.0 e como os experimentos são de caráter exploratório, foi utilizada a configuração padrão fornecida pelo software em todos eles.

Para o algoritmo SMO vale ressaltar que não foi utilizado calibrador (buildCalibrationModels = false), foi utilizado normalização do conjunto de dados (filterType = Normalize training data) e foi Utilizado kernel polinomial de grau 1.

Já para o MLP foi utilizado o número resultante da conta metade da soma do número de atributos com o número de classes de (hiddenLayers = ‘a’), também foi utilizada a normalização de atributos, quinhentas épocas (trainingTime = 500) e taxa de aprendizado igual a 0,3.

As configurações completas são ilustradas no Apêndice A.

3.2 NETBEANS

O NetBeans[2] é um IDE (*Integrated Development Environment* ou traduzindo ambiente de desenvolvimento integrado) multiplataforma, de código aberto, desenvolvido completamente em Java, auxiliando no desenvolvimento de softwares tanto Empresariais quanto Mobile, *Desktop*, Web. Ele conta com suporte a várias linguagens, sendo algumas delas PHP, Java, Javascript, C#. Desde que mudou para código aberto a comunidade de desenvolvedores que contribuem em sua manutenção não parou de crescer, o tornando uma das IDE's mais populares atualmente (OFICINA DA NET, 2008).

Essa IDE possui uma ótima documentação, inclusive traduzida para o português, além de um conjunto de rotinas e bibliotecas que tornam mais ágil o desenvolvimento. Além das funcionalidades mais comuns como por exemplo, copiar, colar, depurar, auto completar, ele ainda conta com ferramentas de refatoração, e geração de código, buscas por referências, entre outros recursos, o que pode impactar no tempo de desenvolvimento, pois possibilita ao desenvolvedor se concentrar mais na lógica do que na implementação de métodos (TOSIN, 2016).

3.3 WAMP

Para desenvolvimento PHP (PHP: *Hypertext Preprocessor*), normalmente é utilizado o Apache como servidor de aplicação, o PHP como interpretador e compilador e banco de dados MySQL. Todos podem ser instalados separadamente em inúmeros sistemas operacionais, sendo Windows ou Linux os mais utilizados (GLASS et. al., 2004).

O Wamp[3] é um software utilizado para facilitar esse processo, pois sua instalação contempla os três softwares citados, além do PhpMyAdmin, utilizado para gerenciar bancos de dados MySQL. O nome Wamp provêm da junção das iniciais de Windows, Apache, MySQL e PHP. Utilizando esse software é possível ter um ambiente de desenvolvimento Web completo sem ao menos abrir os arquivos de

---

[2] https://netbeans.org/
[3] http://www.wampserver.com/en/

configuração, possibilitando que o foco seja o desenvolvimento, e não a configuração dos softwares. Porém, por ele ter os três principais servidores utilizados para o desenvolvimento em PHP, ele possibilita que sejam alteradas quaisquer configurações que seriam alteradas em um servidor convencional.

Quando em execução, ele mantém um ícone na barra de tarefas do Windows, no qual é possível ter acesso as principais funcionalidades do programa, como inicializar os serviços ou reiniciá-los, alterar suas configurações, entre outras funcionalidades.

3.4 WEKA

Desenvolvido pela Universidade de Waikato, localizada nas ilhas da Nova Zelândia e batizado com o nome de um pássaro local, o WEKA[4] (Waikato Environment for Knowledge Analysis), é desenvolvido em Java e tem licença GNU (GNU is Not Unix) General Public License (GPL). Ele é um pacote de aplicativos cuja principal utilidade é a mineração de dados. O WEKA conta com algoritmos de pré-processamento dos dados e de aprendizado de máquina, os quais são utilizados para classificação, agrupamento, regressão, associação e seleção de atributos, além disso ele possibilita a análise dos dados em forma de gráficos e listas (WEKA, 2016).

Abaixo será relatado sobre alguns de seus algoritmos.

- NaiveBayes: esse algoritmo é uma das implementações do Aprendizado Bayesiano, mais especificamente o classificador Bayesiano Ingênuo;
- J48: esse algoritmo é a implementação de uma Árvore de Decisão;
- MultilayerPerceptron (MLP): implementação da uma RNA, mais especificamente a Perceptron Multicamada;
- SMO: implementação da lógica das Maquinas de Suporte de Vetores;
- CfsSubsetEvaltt: esse algoritmo é utilizado para fazer a seleção de atributos em um conjunto de dados. Para isso ele avalia um subconjunto de dados considerando o grau de redundância e a capacidade de

[4] http://www.cs.waikato.ac.nz/ml/weka/

predição de cada atributo. Ele prefere os subconjuntos que tem alta correlação com a classe, tendo baixa intercorrelação;

3.5 LEMATIZADOR DO NILC

O NILC[5] (Núcleo Interinstitucional de Linguística Computacional) foi criado a princípio pela USP (Universidade de São Paulo), em 1993 com a finalidade de promover projetos voltados para a linguística computacional e o processamento de linguagem natural. Hoje conta com cientistas, linguistas e outros pesquisadores de várias universidades e centros de pesquisa.

O NILC disponibiliza várias ferramentas voltadas para a linguística computacional, sendo o lematizador[6] uma delas. Segundo Lucca e Nunes (2002) ele tem por finalidade representar no infinitivo os verbos e no masculino e singular os substantivos, adjetivos, entre outros.

A Lematização é bastante utilizada pelos robôs de busca da Web, pois com ela é possível abranger todos os gêneros, números e tempos verbais utilizados. Na mineração de textos ela é utilizada para agrupar as várias instâncias utilizadas do mesmo objeto, assim aprimorando o resultado final do classificador.

3.6 CORPUS

A empresa Guten Tecnologia forneceu os dados para este estudo, os entregando na forma de um corpus e um arquivo contendo o nível de cada texto, no qual a terceira série é o primeiro nível (1) e a sétima é o último (5). Esse arquivo se chama ids.csv, e contém um número de identificação para cada texto do corpus, sua localização e sua classe. A Figura 11 ilustra um trecho desse arquivo.

---

[5] http://www.nilc.icmc.usp.br/nilc/index.php
[6] http://conteudo.icmc.usp.br/pessoas/taspardo/LematizadorV2a.rar

```
738,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_agua_livro1.txt,5
739,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_alimentos_livro1.txt,2
740,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_alimentos_livro2.txt,1
741,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_anchieta_livro2.txt,1
742,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_animais_livro2.txt,2
743,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_ar_livro1.txt,1
744,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_autoridades_livro1.txt,2
745,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_bandeira_livro1.txt,1
746,Córpus/Corpus_1/corpusNilc/4ano_fund1/nivel2_4ano/nilc_bandeira_livro2.txt,2
```

Figura 11 - Trecho do arquivo ids.csv

Fonte: Autoria Própria

O corpus contém 3811 textos, sendo 1448 já classificados e 2363 sem classe. A Tabela 2 ilustra a distribuição dos textos já classificados no corpus.

Tabela 2 - Detalhamento do corpus

Classe	Nível	Textos
1	3ª Série	242
2	4 ª Série	311
3	5 ª Série	337
4	6 ª Série	285
5	7 ª Série	273

Fonte: Autoria própria

4 EXPERIMENTOS, RESULTADOS E DISCUSSÃO

Neste capítulo serão relatados os experimentos realizados e seus resultados serão apresentados e discutidos. Ele está separado em duas seções, onde na primeira são relatados os experimentos inicias, que embasaram a proposta de experimentos refinados, visando entender os requisitos e os recursos necessários. Na segunda seção são relatados os experimentos refinados.

4.1 EXPERIMENTOS INICIAIS

Primeiramente foram feitas a instalações do Java, Wamp, WEKA e Netbeans para que os experimentos pudessem ser realizados.

```
$p = fopen ($arquivo,"r");
$wr = fopen($narq, "w");
$ln = "";

while (!feof ($p)) {
    $lin = " ".fgets($p,4096);//Lê uma linha do arquivo

    $lin = strip_tags($lin);//Remove tag HTML

    $lin = preg_replace("/ [^a-zA-ZÀ-Úà-ú ] /", "", $lin);
    $lin = preg_replace("/[^a-zA-ZÀ-Úà-ú ]/", " ", $lin);

    $lin .= PHP_EOL;

    $lin = utf8_decode($lin);

    $lin = str_replace("?", "", $lin);
    $ln .= $lin;
    $escreve = fwrite($wr, $lin);

}
fclose($p);
fclose($wr);
$cmd = "java -jar lematizador.jar $narq nf";
$output = shell_exec($cmd);
```

Código 2 - Trecho do algoritmo de pré-processamento

Fonte: Autoria Própria

Após isso, uma rotina em PHP foi utilizada para efetuar a leitura do arquivo ids.csv, o qual contém o número de identificação, localização e classe de cada texto. Para cada texto do corpus, a rotina efetua a leitura e o pré-processa. Nessa fase, o arquivo é convertido para codificação ANSI (*American National Standards Institute*),

pois o lematizador utilizado não trabalha com UTF-8 (8-bit *Unicode Transformation Format*), codificação original do corpus. Também foram removidos numerais e caracteres especiais que não fazem partes das palavras. Após isso, essa rotina chama o lematizador do NILC para processar o texto. O Código 2 ilustra o trecho do algoritmo responsável por fazer o pré-processamento do corpus. Esse processo é repetido para cada linha do arquivo CSV. Essa fase do experimento durou aproximadamente 24 horas para processar os textos que já tinham uma classe definida.

Após esse pré-processamento, haviam quatro tipos de arquivo dentro do corpus. Arquivos de texto (.txt), os quais foram utilizados como entrada para o lematizador, e arquivos com extensão mxp (versão pré-processada para lematização), tagged (versão com as etiquetas contendo as classes gramaticais) e out (versão lematizada do texto), gerados como retorno do software.

Para dar sequência aos experimentos foi clonado o corpus gerado e removidos os resíduos, ou seja, os arquivos com extensão txt, mxp e tagged. Após isso foi executada a linha de comando presente no Quadro 1 para invocar um método do aplicativo WEKA que gera um arquivo ARFF com as informações do corpus. Ela invoca o conversor TextDirectoryLoader, recebendo um diretório, onde seus subdiretórios são as classes e os arquivos pertencentes a eles são as instâncias, e como saída ele retorna um arquivo com a extenção .arff.

```
java -cp "\Program Files\WEKA-3-8\weka.jar" \
weka.core.converters.TextDirectoryLoader \
            -dir corpus_out\ >corpus.arff
```

Quadro 1 - Comando para geração do ARFF

Fonte: Autoria Própria

O arquivo ARFF foi aberto pelo WEKA e processado pelo algoritmo StringToWordVector para que fosse gerada uma *bag of words*, a qual foi submetida aos classificadores SMO e MLP, ambos com os parâmetros padrão fornecidos. O SMO obteve uma taxa de acerto médio de 25,13%, porém o MLP não conseguiu concluir sua execução, pois após 2 semanas de processamento sem concluir sua primeira fase, a qual gera o classificador baseado no conjunto, teve sua execução cancelada.

Concluindo a primeira fase de testes foi utilizado o recurso Select Atributes do WEKA, o qual lista os atributos mais relevantes presentes no conjunto de dados, e

possibilita exportar o resultado, assim reduzindo o número de atributos na *bag of words*. Para tal, foi utilizado o algoritmo CfsSubsetEval. Foram processados os dois mesmos classificadores, os quais atingiram uma taxa de acerto de médio de 29%.

Em um segundo teste efetuado, foi adicionado no pré-processador de textos uma função para remover as *stopwords,* além de outros pequenos ajustes a fim de tornar o processamento mais ágil. Com as alterações foi possível aumentar a acurácia do classificador SMO para 32% após selecionar os atributos, sendo melhor para o conjunto de teste do que os obtidos anteriormente, mais ainda não chegando ao resultado de 47% do classificador oficial da Guten, já obtido pela empresa que cedeu os dados, porém que usa outros atributos.

A Figura 12 ilustra a matriz de confusão do experimento realizado utilizando o classificador SMO e a base de dados após a seleção de atributos, o qual atingiu 32% de acurácia. Pode-se perceber que o a taxa de acerto no nível 1 é maior do que a realizada nos outros níveis. Também é possível perceber um desbalanceamento dos atributos resultantes após a seleção de atributos tendendo para o nível 1.

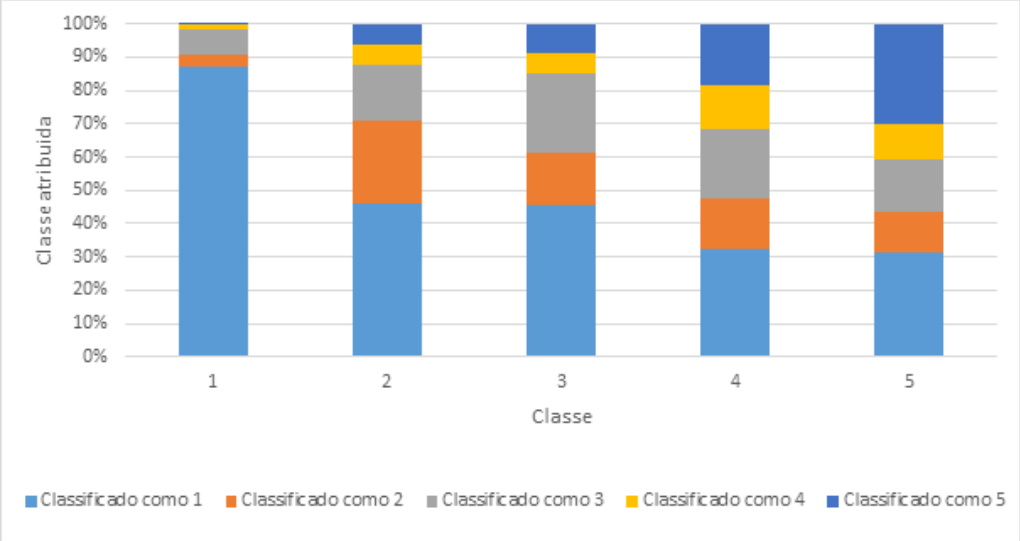

Figura 12 - Gráfico de confusão - SMO com atributos reduzidos

Fonte: Autoria Própria

Esses resultados não foram satisfatórios, pois além de não obter um acerto próximo ao obtido pela empresa, a grande maioria do corpus foi classificado em uma classe distante da esperada, enquanto o classificador da empresa que atinge 47%, tem seus itens classificados nas classes próximas quando os classifica incorretamente (por exemplo, quando um texto do nível 2 é classificado como nível 1). Isso que levou

a uma análise do motivo de não terem atingido uma taxa de acerto mais alta, ou uma classificação mais próxima as classes esperadas. Primeiramente foi analisado o lematizador do NILC, e foi constatado que o mesmo deveria ser substituído, pois ele não possui a acurácia necessária para que o resultado esperado possa ser atingido. Um exemplo é a frase “eu canto mal”, onde ele deveria ter substituído a palavra “canto” pelo verbo “cantar”, mas interpretou essa palavra como se fosse um substantivo, assim não a alterando, pois, já está no masculino e no singular para o caso de um substantivo.

4.2 EXPERIMENTOS REFINADOS

Para uma melhor organização do texto os experimentos refinados foram divididos da seguinte forma:

- Preparação do ambiente.
- Pré-processamento.
- Experimento 01;
- Experimentos 02 a 11;
- Experimento 12;
- Experimento 13;

Nas seguintes subseções eles serão descritos.

4.2.1 Preparação do ambiente

Para que fosse possível a realização do projeto foi definido que as informações utilizadas para o processamento seriam salvas em uma base da dados. A Figura 13 ilustra a estrutura do banco de dados gerado. Os campos mais importantes estão presentes na tabela corpus, sendo os campos id, classe, txt_origem, txt_lematizado e txt_luhn. Os outros campos presentes na tabela corpus são secundários, utilizados como auxiliares no processamento e para análises posteriores. A tabela palavras é uma tabela auxiliar para o corte de Luhn e na tabela delaf_pb2 é armazenado o dicionário utilizado para a lematização.

Após a criação da base de dados, foi criada uma rotina em PHP para importar os dados disponibilizados. Nessa etapa foram importados 3811 textos, sendo

1448 já classificados e 2363 sem classe. Essa abordagem foi utilizada tendo em vista que os dados ainda passariam pelo processo de lematização e corte de Luhn.

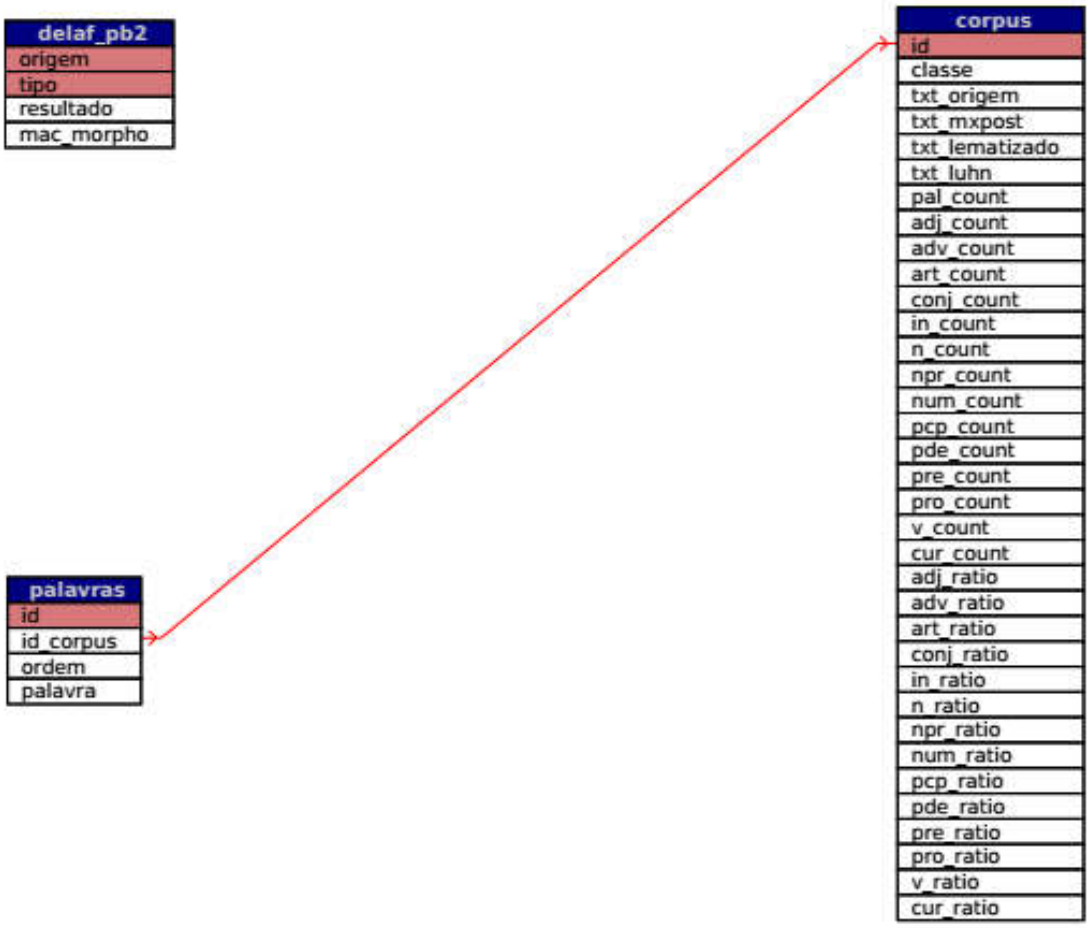

Figura 13 - Esquema Relacional do Banco de Dados

Fonte: Autoria Própria

Após serem importadas as informações do corpus, também foi importado o dicionário de palavras flexionadas (DELAF) encontrado no site do NILC[7]. Esse dicionário utiliza o padrão DELA (*Dictionnarie Electronique du LADL*), formato desenvolvido pelo LADL (*Laboratoire d'Automatique Documentaire et Linguistique*). O arquivo possui 9.072.338 linhas e o seguinte formato: "palavra,canônica.classe+traços:flexão". Foram importadas apenas as informações das palavras, suas respectivas canônicas e classes. A rotina de importação

[7] http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html

desenvolvida remove todos os itens da tabela delaf_pb2 e os insere novamente, a cada importação, totalizando 8.923.205 linhas no banco de dados. A diferença entre o número de linhas presentes no dicionário de palavras e na base de dados existe devido a algumas palavras, aparecerem mais de uma vez com a mesma canônica e classe, porém com flexões diferenciadas. Um exemplo do ocorrido acontece nas linhas treze (ababada,ababadar.V:P3s) e quatorze (ababada,ababadar.V:Y2s) do arquivo, ambas possuem a palavra “ababada”, sua canônica “ababadar” e ambas são verbos, porém a primeira está na terceira pessoa do presente do indicativo e a segunda está na segunda pessoa do imperativo. Como a flexão não é utilizada nos experimentos, a mesma foi ignorada, importando apenas uma vez cada combinação de palavra + canônica + classe.

4.2.2 Pré-processamento

O lematizador utilizado anteriormente levava aproximadamente 24 horas para processar os 1.448 textos já classificados no corpus, enquanto o desenvolvido, em 30 minutos conseguiu lematizar os mesmos 1.448 itens. Para que isso fosse possível, foi utilizado o processamento paralelo, em conjunto com o navegador. Primeiramente a rotina lematizador.php busca o número de identificação de todos os itens que deverão ser lematizados e gera um *javascript* que abre em um pop-up a rotina que lematiza cada texto. Como a rotina que lematiza os textos é independente, pode ter mais de um item sendo processado paralelamente. Após vários experimentos foi constatado que em sua maioria, cada item é processado em aproximadamente 1 segundo, tempo que foi definido entre a chamada de cada item.

O processo da rotina lematizar.php segue o fluxo ilustrado pela Figura 14, sendo:

- Leitura do texto armazenado no banco de dados: é efetuada a leitura do texto referente ao número de identificação recebido por parâmetro;
- Pré-processamento: nessa etapa são removidas as etiquetas HTML (*HyperText Markup Language*) do texto e adicionado espaço entre as palavras e os demais caracteres;
- Armazenamento do arquivo pré-processado: é salvo em disco, com a extenção .txt, para ser utilizado na etapa de etiquetagem;

- Etiquetagem: é utilizado nessa etapa o *tagger* MX-POST[8], junto com o dicionário mac-morpho[9], que recebe o texto de origem e retorna cada palavra do texto com uma etiqueta, a qual marca cada palavra com sua devida classe gramatical;

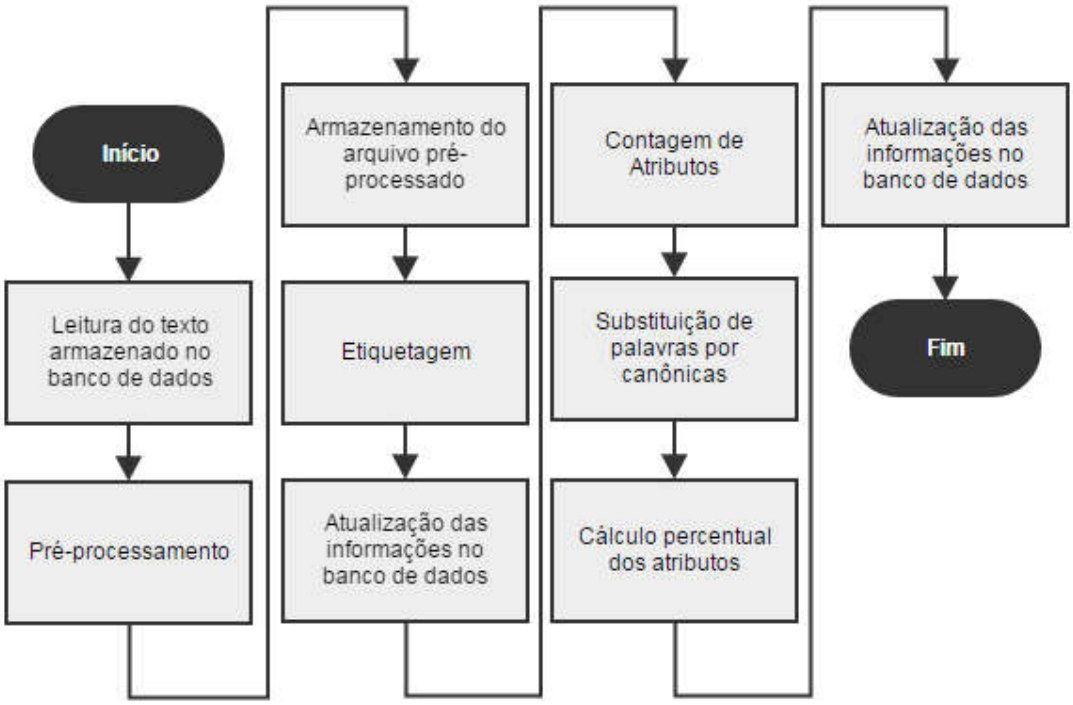

Figura 14 - Fluxograma da rotina lematizar.php

Fonte: Autoria Própria

- Atualização de informações no banco de dados: é efetuada a leitura do arquivo gerado pelo *tagger* e são atualizadas no banco de dados as informações contidas nesse arquivo;
- Contagem de Atributos: nessa etapa é contado o número de palavras do texto e também o número de palavras por classe gramatical, de acordo com os dados gerados no processo de etiquetagem;
- Substituição de palavras por canônicas: foi substituída cada palavra etiquetada por sua respectiva canônica, levando em consideração a etiqueta gramatical gerada no processo de etiquetagem;

[8] http://www.statmt.org/moses/?n=Moses.ExternalTools
[9] http://www.nilc.icmc.usp.br/nilc/download/mxpost-files_macmorpho.zip

- Cálculo percentual dos atributos: nessa etapa é calculado o percentual de cada classe gramatical presente no texto;
- Atualização de informações no banco de dados: é atualizado no banco de dados as informações referentes a contagem de atributos e seus devidos percentuais, além do texto lematizado;

A maior dificuldade até essa etapa foi a diferença entre as etiquetas geradas pelo *tagger* na fase de etiquetagem e as classes da base de dados DELAF. Após analisar os manuais de ambos, foi utilizada a Tabela 3 para fazer a conversão entre suas etiquetas.

Tabela 3 - Equivalência de etiquetas entre os recursos

Descrição	DELAF-PB	mac-morpho
Adjetivo	A	ADJ
Advérbio	ADV	ADV; ADV-KS; ADV-KS-REL
Conjunção	CONJ	KC; KS
Artigo Definido	DET+Art+Def	ART
Artigo Indefinido	DET+Art+Ind	ART
Numeral	DET+Num	NUM
Interjeição	INTERJ	IN
Substantivo	N	N
Nome Próprio	N+Pr	NPROP
Prefixo	PFX	
Preposição	PREP	PREP
Preposição	PREPXADV	PREP
Preposição	PREPXDET+Art+Def	PREP
Preposição	PREPXDET+Art+Ind	PREP
Preposição	PREPXDET+Dem	PREP
Preposição	PREPXDET+Ind	PREP
Preposição	PREPXPREP	PREP
Preposição	PREPXPRO+Dem	PREP
Preposição	PREPXPRO+Ind	PREP
Preposição	PREPXPRO+Int	PREP
Preposição	PREPXPRO+Pes	PREP
Preposição	PREPXPRO+Rel	PREP
Pronome	PRO+Dem	PRO-KS; PRO-KS-REL; PROSUB
Pronome	PRO+Ind	PRO-KS; PRO-KS-REL; PROSUB
Pronome	PRO+Int	PRO-KS; PRO-KS-REL; PROSUB
Pronome	PRO+Pes	PROPESS
Pronome	PRO+Pos	PRO-KS; PRO-KS-REL; PROSUB
Pronome	PRO+Rel	PRO-KS; PRO-KS-REL; PROSUB
Pronome	PRO+Tra	PRO-KS; PRO-KS-REL; PROSUB

Descrição	DELAF-PB	mac-morpho
Pronome	PROXPRO+DemXInd	PRO-KS; PRO-KS-REL; PROSUB
Pronome	PROXPRO+PosXTra	PRO-KS; PRO-KS-REL; PROSUB
Verbo	V	V; VAUX
Verbo	V+PRO	V; VAUX

Fonte: Autoria Própria

Após esse processo, os textos passam pelo corte de Luhn, onde eles são separados palavra a palavra, para que possa ser calculado o número de palavras distintas presentes no corpus e removidas as 10% que menos aparecem e as 10% que mais aparecem. Concluído esse processo, a tabela “corpus” do banco de dados fica preenchida com todas as informações processadas, faltando apenas serem exportadas para análise no WEKA. A Tabela 4 ilustra como ficaram as informações referente ao texto original, e suas modificações após a etiquetagem, lematização e corte de Luhn.

Tabela 4 - Informações após o processamento

txt_origem	<title>Dia do Trabalho</title>No dia 1o de maio homenageamos o trabalhador.Todas as pessoas têm direito a um trabalho ou a uma profissão.A criança que estuda está se preparando para uma futura profissão.
txt_mxpost	dia_N do_N trabalho_N no_PREP dia_N 1o_N de_PREP maio_N homenageamos_V o_ART trabalhador_N ._. todas_PROADJ as_ART pessoas_N têm_V direito_N a_PREP um_ART trabalho_N ou_KC a_PREP uma_ART profissão_N ._. a_ART criança_N que_PRO-KS-REL estuda_V está_V|+ se_PROPESS preparando_V para_PREP uma_ART futura_ADJ profissão_N ._.
txt_lematizado	dia dó trabalho no dia NUM de maio homenagear o trabalhador todo o pessoa ter direito a um trabalho ou a um profissão o criança que estudar estar ele preparar para um futuro profissão
txt_luhn	maio homenagear trabalhador profissão preparar futuro profissão

Fonte: Autoria Própria

O próximo procedimento que foi realizado foi desenvolver uma rotina para que o corpus armazenado no banco de dados fosse exportado. O corpus foi exportado em um diretório nomeado “corpus”, na raiz do projeto, e para cada classe de texto foi gerado um subdiretório onde os textos respectivos a essa classe foram armazenados. Ao concluir a exportação do corpus, a própria rotina desenvolvida executa dois comandos utilizando a função “exec” do PHP, a qual roda os comandos diretamente no *prompt* de comando do Sistema Operacional. O primeiro executa o conversor TextDirectoryLoader, e o segundo o filtro StringToWordVector, ambos pertencentes ao WEKA.

4.2.3 Experimento 01

Assim como nos experimentos iniciais foram utilizados os classificadores SMO e MLP para classificar os dados processados, utilizando suas configurações padrão. Novamente como nos experimentos iniciais, o classificador MLP não conseguiu concluir seu processamento. A hipótese mais provável é que devido ao grande número de atributos existentes na base de dados (por volta de dezesseis mil), seja necessário um equipamento com maior poder de processamento, durante o tempo em que o experimento rodou, não foi possível nem mesmo terminar o primeiro *fold*, na indução da validação cruzada baseada em 10 *folds*, também escolhida por ser padrão do WEKA. Já o classificador SMO teve uma acurácia de 24,31%, um resultado insatisfatório, tendo em vista que nos experimentos iniciais já havia sido atingido um resultado de 32%.

Para que fosse possível melhorar o resultado, foi utilizado o recurso de seleção de atributos, e após isso foram processados os dados gerados nos dois classificadores. Com isso foi possível atingir um acerto de 25,35% no MLP e 26,31% no SMO.

4.2.4 Experimentos 02 a 11

Como os resultados ainda não haviam sido satisfatórios, foi analisado o resultado das etapas do processamento. Foi constatado que o corte de Luhn poderia ser o problema, pois há inúmeras palavras, que mesmo após a lematização, aparecem apenas uma vez no corpus. Devido a isso foi feito uma nova rodada de experimentos, onde, antes de efetuar o corte no corpus, eram removidas todas as palavras que apareciam apenas 1 vez. O resultado teve uma pequena melhora, sendo 25,48% utilizando o classificador SMO, e novamente o MLP não conseguiu concluir sua execução. Após a seleção de atributos o SMO atingiu a acurácia de 26,11% e o MLP de 24,79%.

Constatada uma leve melhora nos resultados andes da seleção de atributos, foram executados mais 9 experimentos semelhantes, onde foram removidas as palavras que apareciam apenas duas, três, quatro, e assim por diante. Os resultados dos 11 experimentos realizados são ilustrados na Tabela 5.

Analisando os dados contidos na tabela, pode-se perceber uma leve melhora nos resultados antes da Seleção de Atributos, de acordo com a redução no

número de atributos, porém, após ela ocorre o inverso. Um fato interessante é que no Experimento 11, antes da Seleção de Atributos, foi obtido uma acurácia próxima a do Experimento 1 após esse processo.

Tabela 5 - Resultados Corte de Luhn

Experimento	Aparições das Palavras Removidas	Corte de Luhn			Seleção de Atributos		
		Atributos	SMO	MLP	Atributos	SMO	MLP
1	Sem	12.686	24,31%	-	19	26,31%	25,35%
2	1 Vez	7.060	25,48%	-	19	26,11%	24,79%
3	2 Vezes	5.037	24,45%	-	20	26,31%	25,48%
4	3 Vezes	4.108	25,14%	-	21	26,93%	26,04%
5	4 Vezes	3.471	25,48%	-	19	26,66%	25,90%
6	5 Vezes	3.057	25,69%	-	19	26,52%	25,07%
7	6 Vezes	2.717	26,93%	-	19	26,52%	25,07%
8	7 Vezes	2.460	25,83%	-	17	26,24%	25,14%
9	8 Vezes	2.252	25,55%	-	15	26,24%	24,59%
10	9 Vezes	2.075	26,93%	-	14	25,69%	24,79%
11	10 Vezes	1.925	26,17%	-	13	25,28%	23,41%

Fonte: Autoria Própria

4.2.5 Experimento 12

Analisando os dados do Pesquisador da Guten, Natan Siegle Hartman, o qual está envolvido no projeto que já havia conseguido os 47% de acurácia, foi constatado que alguns atributos calculados geravam um enorme ganho de desempenho. Por exemplo, utilizando apenas o percentual de adjetivos na base de dados dele foi possível atingir uma média de acerto de 40%. Devido a esse fato foi exportada a informação relativa ao percentual de cada classe gramatical e o número de palavras contidas nos textos.

A expectativa com essa abordagem era que o resultado ficasse mais próximo ao obtido com os dados analisados, porém foi obtido apenas 23,20% de acurácia com o SMO e 21,82% com o MLP. Para tentar melhorar esse resultado foi efetuado a seleção de atributos nessa base e o resultado obtido foi de 22,85% com o SMO e 22,65% com o MLP.

Vale ressaltar que o Pesquisador da Guten, Natan Siegle Hartman, utiliza uma metodologia diferente para a geração dos atributos. Em seus experimentos ele utiliza o etiquetador licenciado “Analisador Sintático Palavras para Tagging e outras operações” além de vários cálculos complexos para a obtenção dos atributos.

4.2.6 Experimento 13

Como o resultado do Experimento 12 independia do corte de Luhn, foi efetuado um novo experimento utilizando apenas o texto lematizado. Utilizando apenas essa informação foram gerados 15967 atributos distintos, e obtido uma acurácia de 25,89% com o classificador SMO, resultado inferior apenas aos obtidos nos Experimentos 10 e 11 antes da Seleção de Atributos. Assim como nos experimentos anteriores, com a exceção do Experimento 12, o MLP não conseguiu concluir seu processamento.

Após a seleção, o número de atributos reduziu para 41 e foi obtida uma acurácia de 28,54% pelo SMO e de 25,41% no MLP. Esse experimento foi o que obteve maior acurácia utilizando o algoritmo SMO, porém inferior a atingida nos experimentos iniciais, indicando que no corte de Luhn atributos relevantes estavam sendo removidos. Devido a isso, foi analisado novamente o lematizador. Ao analisar as etapas do processo foi constatado que o processo de etiquetagem não estava trazendo o resultado esperado. Por exemplo, na seguinte frase, “Ele é um mamífero quadrúpede assim como a vaca” que teve o seguinte retorno do etiquetador: “ele_PROPESS é_V um_ART mamífero_N quadrúpede_V assim_ADV como_KS a_ART vaca_N” a palavra quadrúpede foi etiquetada como um verbo, porêm ela é um adjetivo. Outro ponto que foi percebido foi que alguns textos estavam incompletos após a etiquetagem.

4.3 DISCUSSÃO

Para obter uma análise crítica dos resultados, vale ressaltar alguns pontos:

- Apenas no Experimento 12 foi possível concluir o processamento do algoritmo MLP na primeira fase de processamento dos dados (antes da seleção de atributos), tendo em vista o número reduzido de atributos gerados para esse experimento;
- O corpus possui um leve desbalanceamento, sendo que 17% dos textos pertencem ao nível 1, 21% ao nível 2, 23% ao nível 3, 20% ao nível 4 e 19% ao nível 5;

- A acurácia mínima aceitável para os experimentos é de 23%, pois levando em consideração que a classe majoritária atinge 23% do corpus;

Como convenção, os experimentos serão citados nesta seção da seguinte forma:

- Experimento n (a) – Experimento utilizando o classificador SMO no conjunto de dados completo;
- Experimento n (b) – Experimento utilizando o classificador MLP no conjunto de dados completo;
- Experimento n (c) – Experimento utilizando o classificador SMO no conjunto de dados reduzido;
- Experimento n (d) – Experimento utilizando o classificador MLP no conjunto de dados reduzido;

A Figura 15 ilustra um gráfico referente a matriz de confusão do Experimento 01 (a). Pode-se perceber uma distribuição uniforme das classes, revelando que os atributos contidos não são muito significantes para o classificador, resultando em um aprendizado pobre.

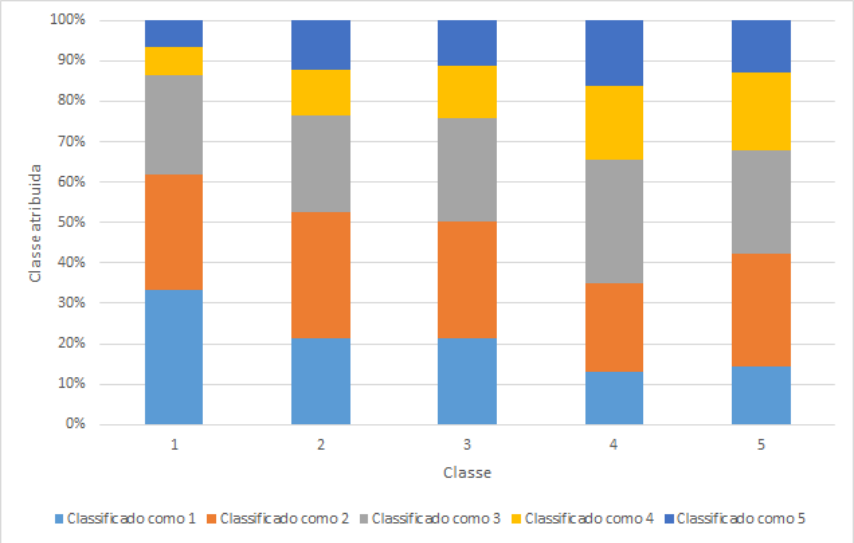

Figura 15 - Gráfico de confusão - Experimento 01 (a)

Fonte: Autoria Própria

Já a Figura 16, a qual ilustra o gráfico de confusão referente ao Experimento 01 (c) , demonstra um desbalanceamento nos atributos resultantes após a seleção de atributos, visto que 92% do conjunto foi classificado como se fosse do

nível 3. Devido a isso é possível concluir que 97% da acurácia atingida nesse experimento (26,31%) é referente ao nível 3.

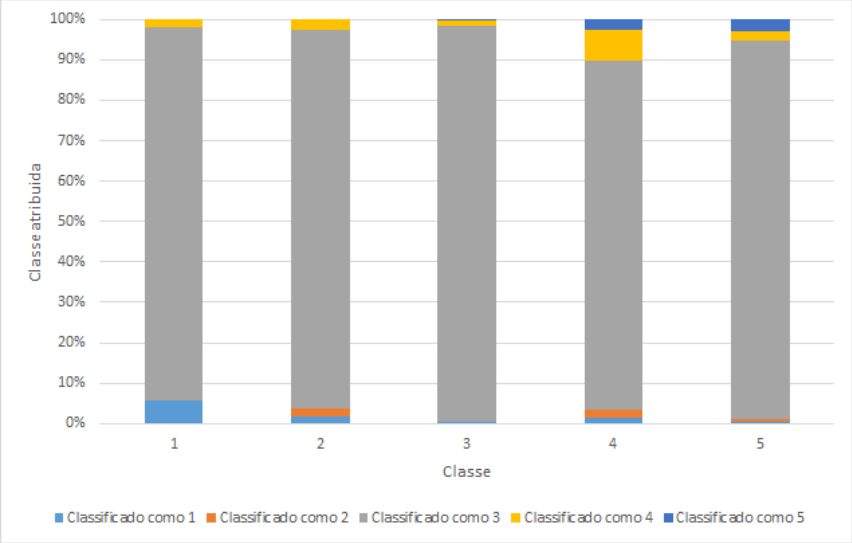

Figura 16 - Gráfico de confusão - Experimento 01 (c)

Fonte: Autoria Própria

Diferente das figuras acima, na Figura 17 é ilustrada a matriz de confusão do Experimento 01 (d), o qual foi realizado utilizando o classificador MLP e o conjunto reduzido. Assim como no Experimento 01 (c) é possível ver uma maior concentração das hipóteses no nível 3, porém com uma distribuição menos concentrada do que com o classificador SMO.

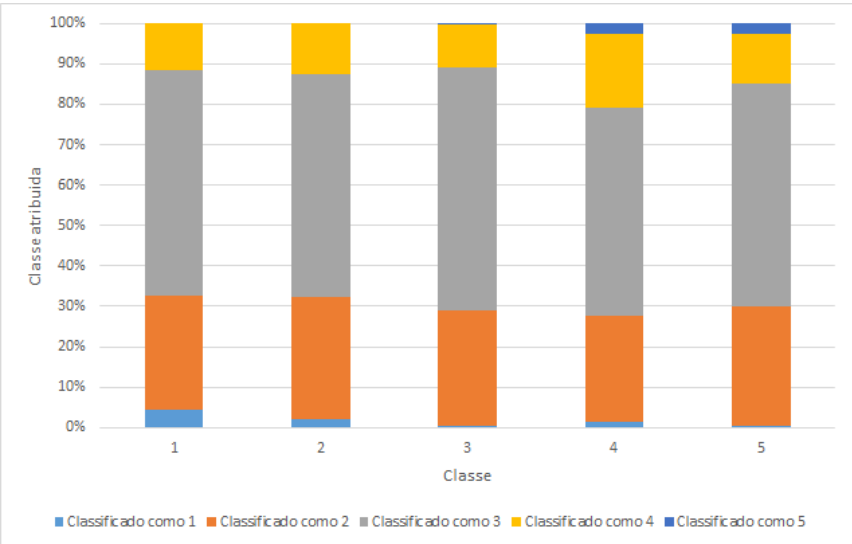

Figura 17 - Gráfico de confusão - Experimento 01 (d)

Fonte: Autoria Própria

Situação semelhante a ilustrada nas figuras acima, também acontece em todos os experimentos, sendo que nos experimentos utilizando o conjunto completo existe uma distribuição uniforme dos itens classificados, indicando que os atributos fornecidos como entrada não eram o suficiente para uma taxa de acerto mais próxima dos 47% já atingidos pelo classificador oficial da Guten. Nos experimentos utilizando o conjunto de dados reduzido os itens foram classificados em sua maioria como se fossem do nível 3, indicando o desbalanceamento dos atributos e resultando em um aprendizado tendencioso.

Em relação a piora no resultado com o novo lematizador desenvolvido, pode ser atribuído a uma falha no etiquetador MX-POST, levando em consideração, que o pré-processamento dos textos foi semelhante, em ambos os casos.

5 CONSIDERAÇÕES FINAIS

Nesse capítulo serão apresentadas as conclusões obtidas após a realização desse trabalho, além das alterações e melhorias que podem ser implementadas a fim de aprimorar os resultados.

5.1 CONCLUSÃO

Nesse trabalho foi utilizado um corpus cedido pela empresa Guten Tecnologia, em conjunto com algumas ferramentas gratuitas disponíveis para fazer parte do pré-processamento e classificação dos textos. Utilizando os classificadores SMO e MLP não foi possível melhorar a taxa de acerto atingida anteriormente pela empresa.

Foi possível apenas aprimorar o desempenho da fase de pré-processamento de textos, porém não foi possível obter uma base de dados confiável com as ferramentas utilizadas (Lematizador do NILC, MX-POST, mac-morpho).

Os objetivos do trabalho foram cumpridos, porém não trazendo um resultado positivo ao experimento. Em decorrência dos resultados negativos obtidos, não foi possível utilizá-los no projeto referido durante o Capítulo 1, porém os classificadores gerados podem ser utilizados como *baseline* para outros experimentos.

5.2 TRABALHOS FUTUROS / CONTINUAÇÃO DO TRABALHO

É proposto uma continuação desse trabalho, uma análise em um Córpus maior, podendo distinguir com maior clareza se os resultados negativos são exclusivamente uma falha no pré-processamento, ou também foi influenciado pelo tamanho do Córpus. Também pode ser interessante trabalhar o lematizador podendo obter uma melhor acurácia. Outros atributos ainda podem ser explorados com grande potencial de resultados melhores.

Outra abordagem que pode ser utilizada é a de *word embeddings*, uma representação alternativa em relação a bag of words, nesta representação os vetores são densos ao invés de esparsos.

Tendo resultados mais confiaveis pode ser desenvolvido um ambiente de aprendizado online, possibilitando que as crianças entre a terceira e sétima série do ensino fundamental tenham a possibilidade de ler textos adequados à sua fase de aprendizado.

REFERÊNCIAS BIBLIOGRÁFICAS

ARANHA, Christian Nunes. **Uma Abordagem de Pré- Processamento Automático para Mineração de Textos em Português: Sob o Enfoque da Inteligência Computacional.** 2007. 144 f. Tese (Doutorado) - Curso de Doutorado em Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Disponível em: <http://livros01.livrosgratis.com.br/cp064589.pdf>. Acesso em: 25 nov. 2016.

COPPIN, Ben. **Inteligência artificial.** Grupo Gen-LTC, 2015.

FACELI, Katti et al. **Inteligência Artificial–uma abordagem de aprendizado de máquina.** Rio de Janeiro: LTC, 2011.

GLASS, Michael K. et al. **Beginning PHP, Apache, MySQL Web Development.** John Wiley & Sons, 2004.

HAYKIN, Simon S. **Redes neurais artificiais: princípio e prática.** 2ª Edição, Bookman, São Paulo, Brasil, 2000.

IBGE. **Séries Estátísticas e Séries Históricas.** Disponível em: <http://seriesestatisticas.ibge.gov.br/default.aspx?op=busca>. Acesso em: 26 ago. 2016.

LUCCA, J. L. de; NUNES, Maria das Graças Volpe. **Lematização versus Stemming.** São Carlos, Sp: Nilc - Icmc-usp, 2002. (Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional NILC - ICMC-USP, Caixa Postal 668, 13560-970 São Carlos, SP, Brasil). Disponível em: <http://www.nilc.icmc.usp.br/nilc/download/lematizacao_versus_steming.pdf>. Acesso em: 23 ago. 2016.

MARTINS, Claudia A.; MONARD, Maria Carolina; MATSUBARA, Edson T. **Uma metodologia para auxiliar na seleçao de atributos relevantes usados por algoritmos de aprendizado no processo de classificaçao de textos.** In: XXIX Conferencia LatinoAmericana de Informatica-CLEI, La Paz, Bolıvia.(to be published). 2003.

MITCHELL, Tom M. **Machine learning.** WCB. 1997.

OFICINA DA NET, Redação. **O Que é o NetBeans?** 2008. Disponível em: <https://www.oficinadanet.com.br/artigo/1061/o_que_e_o_netbeans>. Acesso em: 20 jul. 2016.

ROSA, João Luis Garcia. **Fundamentos da inteligência artificial**. Editora LTC, 2011.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência artificial: tradução da terceira edição.** Elsevier, 2013.

SAMUEL, Arthur L. **Some studies in machine learning using the game of checkers.** IBM Journal of research and development, v. 3, n. 3, p. 210-229, 1959.

TOSIN, Carlos Eduardo Gusso. **Explorando o NetBeans 7.0 - Artigo Java Magazine 91.** Disponível em: <http://www.devmedia.com.br/explorando-o-netbeans-7-0-artigo-java-magazine-91/21121#>. Acesso em: 20 jul. 2016.

WEKA. **WEKA 3: Data Mining Software em Java.** Disponível em: <http://www.cs.waikato.ac.nz/~ml/weka/>. Acesso em: 11 ago. 2016.