



ALEXANDRE AUGUSTO ALBERTO MOREIRA DE ABREU

**MÁQUINA DE APRENDIZADO EXTREMO
APLICADA À ANÁLISE DE SENTIMENTOS**

LAVRAS - MG

2012

ALEXANDRE AUGUSTO ALBERTO MOREIRA DE ABREU

**MÁQUINA DE APRENDIZADO EXTREMO APLICADA À ANÁLISE DE
SENTIMENTOS**

Monografia apresentada ao Colegiado do
Curso de Ciência da Computação, para a
obtenção do título de Bacharel em Ciên-
cia da Computação.

Orientador

Prof. Dr. Denilson Alves Pereira

Co-Orientador

Prof. MSc. Tiago Amador Coelho

LAVRAS - MG

2012

ALEXANDRE AUGUSTO ALBERTO MOREIRA DE ABREU

**MÁQUINA DE APRENDIZADO EXTREMO APLICADA À ANÁLISE DE
SENTIMENTOS**

Monografia apresentada ao Colegiado do
Curso de Ciência da Computação, para a
obtenção do título de Bacharel em Ciên-
cia da Computação.

Aprovada em 22 de Outubro de 2012

Profa. Dra. Marluce Rodrigues Pereira

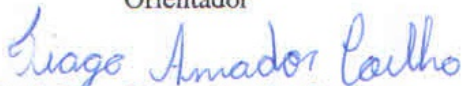


Prof. MSc. Eric Fernandes de Mello Araújo



Prof. Dr. Denilson Alves Pereira

Orientador



Prof. MSc. Tiago Amador Coelho

Co-Orientador

LAVRAS - MG

2012

In memoriam de Virgílio Augusto Moreira de Abreu que, apesar de sua breve estadia entre nós, nos presenteou imensamente com sua alegria e amor.

AGRADECIMENTOS

Especiais agradecimentos a Ana Rosalva Moreira e Aníbal Moreira de Abreu os meus anjos da guarda, Doutor Claudemir de Abreu meu herói, Maya pelo carinho e compreensão, todos os meu familiares e amigos pelo apoio, orientadores e mestres pela iluminação, serviçais, coadjuvantes e antagonistas pois sem vocês minha história não seria a mesma.

RESUMO

O aumento, instantâneo, da quantidade de informações (e opiniões) disponíveis na rede mundial de computadores (*Internet*), alavanca a necessidade da criação e melhoria dos métodos e ferramentas capazes de explorar o conteúdo opinativo e emocional publicados, diariamente, por grande e crescente quantidade de usuários. Técnicas de Inteligência Computacional tem sido largamente usadas em aplicações de Mineração de Dados. Numerosas técnicas de Inteligência Artificial, Redes Neurais Artificiais e *Support Vector Machines (SVMs)* têm representado os papéis principais. Entretanto, sabe-se que ambos, redes neurais e *SVMs*, encaram alguns desafios e problemas como: lentidão de aprendizagem, intervenção humana não trivial e/ou baixa escalabilidade computacional. Máquina de Aprendizado Extremo (*ELM*), um método emergente para treinamento de Redes Neurais Artificiais, surge para sobrepor alguns destes desafios encontrados pelas outras técnicas. *ELM* funciona para *Single-Hidden Layer Feedforward Networks (SLFNs)* e sua essência é de que a camada escondida das *SLFNs* não precisa ser ajustada. Quando comparada com os métodos de Inteligência Computacional, *ELM* provê uma melhor performance de generalização em um tempo muito mais rápido, sem que seja necessária intervenção humana. O presente trabalho pretende avaliar estas e outras características do recente *ELM* com a finalidade de realizar uma análise quantitativa e qualitativa. A intenção é realizar um comparativo entre o método proposto e os métodos *SVM*, *KNN* e *MLP*, para a análise de sentimentos e mineração de opinião.

Palavras-chave: Mineração de Opinião; Análise de Sentimento; Máquina de Aprendizado Extremo; Rede Neural; Mineração de Texto.

ABSTRACT

The instantaneous increase the amount of information (and opinions) available on the World Wide Web (Internet), leverages the need for the creation and improvement of methods and tools able to exploit the opinational and emotional contents published, diary, by a growing amount of users. Computational Intelligence techniques have been used in wide Data Mining applications. Out of numerous Artificial Intelligence techniques, Artificial Neural Networks and Support Vector Machines (*SVMs*) have been played the leading roles. However, it is known that both, Neural Networks and *SVMs*, face some challenging and troubles issues such as: slowness learning speed, not trivial human intervention and/or poor computational scalability. Extreme Learning Machine (*ELM*), an emergent method to training Neural Networks, arrives to overcome some of this challenges faced by other techniques. *ELM* works for Single-Hidden Layer Feedforward Networks (*SLFNs*) and its essence is that the hidden layer of *SLFNs* doesn't need to be tuned. When compared with the traditional computational intelligence techniques, *ELM* provides better generalization performance at a much faster learning speed without requiring human intervention. This study intends to evaluate these and other features of the recent *ELM* in order to perform a quantitative and qualitative analysis. The intention is to conduct a comparative between the proposed method and the *SVM*, *KNN* e *MLP* methods, to the analysis of sentiment and opinion mining.

Keywords: *Opinion Mining; Sentiment Analysis; Extreme Learning Machine; Neural Networks; Text Mining.*

LISTA DE FIGURAS

Figura 1	Processo de <i>KDT</i> adaptado de Aranha et al (2007)	17
Figura 2	Exemplo de classificação usando <i>SVM</i>	30
Figura 3	Esquema de um neurônio adaptado de Vellasco, M. M. B. R. (1994)	36
Figura 4	<i>Single-hidden layer feedforward network</i> adaptado de Vellasco, M. M. B. R. (1994)	38

LISTA DE TABELAS

Tabela 1	Exemplo de representação do Modelo de Espaço Vetorial.	21
Tabela 2	Exemplo de mensagens e suas classes	45
Tabela 3	Tabela de contingência. Adaptado de Lin, Hsieh e Chuang 2009...	46
Tabela 4	Interpretação dos valores de <i>Estatística Kappa</i> . Extraído de Landis e Koch 1977	48
Tabela 5	Distribuição dos dados	50
Tabela 6	Resultados da execução do <i>SVM</i>	59
Tabela 7	Tempo de treinamento (τ) e execução (ϵ) em segundos para o <i>SVM</i>	60
Tabela 8	Resultados da execução do <i>KNN</i>	61
Tabela 9	Tempo de treinamento (τ) e execução (ϵ) em segundos para o <i>KNN</i>	62
Tabela 10	Resultados da execução do <i>MLP</i>	63
Tabela 11	Tempo de treinamento (τ) e execução (ϵ) em segundos para o <i>MLP</i>	64
Tabela 12	Resultados da execução do <i>ELM</i>	65
Tabela 13	Tempo de treinamento (τ) e execução (ϵ) em segundos para o <i>ELM</i>	66
Tabela 14	Comparativo de execução para o melhor acurácia de cada método.....	67
Tabela 15	Comparativo de execução para a média de cada método.	69

SUMÁRIO

1	Introdução	11
1.1	Motivação	11
1.2	Objetivos	13
1.3	Organização do trabalho	13
2	Mineração de Texto	15
2.1	<i>Knowledge Discovery in Textual Database (KDT)</i>	16
2.2	Recuperação de Informação	18
2.3	Indexação	20
2.4	Processamento de Língua Natural	22
2.5	Mineração de Texto, Processamento de Língua Natural e aplicações	24
3	Análise de Sentimento ou Mineração de Opinião	25
3.1	Máquinas de aprendizado (<i>Learning Machines</i>).....	27
3.1.1	<i>Support Vector Machine</i>	28
3.2	Análise de Semântica Latente	29
3.3	Dificuldades.....	31
4	Redes Neurais	33
4.1	Redes Neurais Artificiais	35
4.2	Especificações.....	36
4.3	<i>Feedforward Neural Network</i>	37
4.4	<i>Extreme Learning Machine</i>	39
4.4.1	Funcionamento básico.....	40
4.4.2	Vantagens	42
5	Metodologia de desenvolvimento	44
5.1	Classificação das mensagens	44

5.2	Métricas	45
5.3	Coleta dos dados	49
5.3.1	Método de coleta	51
5.3.2	Critérios adotados para coleta.....	51
5.3.3	Classificação manual das mensagens	52
5.4	Pré-processamento e indexação	53
5.5	Mineração: Análise de Sentimento	54
5.5.1	<i>Support Vector Machine (SVM)</i>	54
5.5.2	<i>K-Nearest Neighbours (KNN)</i>	55
5.5.3	<i>Multilayer Perceptron (MLP)</i>	56
5.5.4	<i>Extreme Learning Machine (ELM)</i>	56
6	Resultados e análise.....	58
6.1	<i>SVM</i>	59
6.2	<i>KNN</i>	61
6.3	<i>MLP</i>	62
6.4	<i>ELM</i>	64
6.5	Discussões gerais	67
7	Conclusões e trabalhos futuros	70

1 Introdução

Neste capítulo inicial, são apresentados de maneira concisa a motivação, os objetivos e a organização do presente trabalho.

1.1 Motivação

O surgimento e, principalmente, o crescimento expressivo da *Internet*, sistemas de busca e redes sociais nas duas últimas décadas tem feito o número de informação textual escrito em linguagem natural atingir proporções estrondosas. Todo esse grande contêiner de informações (e opiniões) se mostra expressivamente instigante para pesquisadores e empresas. Diversos estudos, pesquisas e desenvolvimento surgem com o intuito de explorar esse potencial que representa, atualmente, o maior bem e poder da humanidade: a informação.

Seu grande potencial se deve, não somente ao crescimento contínuo da *Internet*, mas também dos mecanismos de buscas. Durante a década de 90, estimou-se que o crescimento médio anual foi de 100%, com um período de crescimento massivo entre 1996 e 1997 (COFFMAN; ODLYZKO, 1998).

Estimativas apontam que 85% das informações comerciais estão sob a forma textual (TEXTMININGNEWS, 2011). Fica evidente que tanto a iniciativa pública, quanto a privada, têm grande interesse em explorar tais informações, principalmente, no que tange às opiniões dos consumidores. Por exemplo, uma empresa pode estar interessada na aceitação de um produto, serviço ou de modificações realizadas recentemente (em seus produtos e/ou serviços), a fim de tomar decisões estratégicas como:

- Manter o foco na produção de um determinado produto em detrimento de outro.
- Modificar determinada funcionalidade.
- Realizar melhorias, ou um *downgrade* de versão, para determinado *software*.
- Abandonar a produção de determinado produto ou deixar de prestar este ou aquele serviço.
- Outras inúmeras aplicações de *Business Intelligence (BI)*.

Outro exemplo que vem ganhando atenção é o de personalidades públicas - como políticos, astros da televisão ou música - querendo saber a opinião do público sobre sua imagem. Essa análise, pode inclusive, ser realizada sobre o presente momento ou apresentada uma progressão cronológica capaz de demonstrar os momentos em que a opinião esteve mais positiva ou mais negativa. Na seção 2.5 estas questões serão melhores detalhadas e exemplificadas.

Técnicas de Inteligência Computacional tem sido largamente usadas em aplicações que se propõem a realizar estas tarefas em tempo hábil e com boa acurácia. Porém, tanto técnicas de Inteligência Artificial, Redes Neurais e *Support Vector Machines (SVMs)* enfrentam, atualmente, diversas questões e problemáticas que pretende-se contornar com a abordagem aqui proposta. As principais questões que precisam ser contornadas são: lentidão de aprendizagem, intervenção humana não trivial e/ou baixa escalabilidade computacional. As duas primeiras questões impactam diretamente no tempo necessário para se completar tal tarefa, enquanto que a última questão influencia diretamente na qualidade dos resultados. No decorrer deste trabalho tais questões serão melhores detalhadas e soluções propostas.

Máquina de Aprendizado Extremo (*ELM*), um método emergente, surge para minimizar alguns destes desafios encontrados pelas outras técnicas (HUANG *et al.*, 2004). *ELM* funciona para *Single-Hidden Layer Feedforward Networks (SLFNs)* e sua essência é de que a camada escondida das *SLFNs* não precisa ser ajustada.

1.2 Objetivos

Este trabalho tem como objetivo, implementar e utilizar de forma prática *ELM* para realizar Análise de Sentimento (Mineração de Opinião) e realizar um comparativo entre os métodos clássicos da literatura (*SVM*, *KNN* e *MLP*) gerando uma análise quantitativa e qualitativa dos resultados.

1.3 Organização do trabalho

Primeiramente, no capítulo 2 é realizada uma análise de referencial teórico multidisciplinar envolvendo: Mineração de Texto (*Text Mining*), Recuperação de Informação e Processamento de Língua Natural. O capítulo 3 apresenta informações sobre Análise de Sentimento (Mineração de Opinião) e o capítulo 4 descreve Redes Neurais Artificiais e Máquina de Aprendizado Extremo (*Extreme Learning Machine*).

São apresentados fatos e dados motivacionais para o estudo e aplicações de assuntos, como, Máquinas de Aprendizado (*Extreme Learning Machine* especialmente), *SVM*, Processamento de Língua Natural, Mineração de Texto e exemplos práticos de sistemas que fazem uso dessa gama de conhecimentos, métodos e ferramentas.

Na sequência encontra-se o Capítulo 5 referente à metodologia de trabalho utilizada, assim como uma explicação de como os dados foram coletados, pré-processados e classificados.

O Capítulo 6 é responsável pela demonstração, confronto e análise dos dados obtidos pelos experimentos.

Finalizando o presente trabalho temos o Capítulo 7 que traz algumas conclusões, por parte do autor, e a proposição de trabalhos futuros nesta linha de pesquisa.

2 Mineração de Texto

A Mineração de Texto (*Text Mining*, também conhecida como *Intelligent Text Analysis*, *TextData Mining* ou *Knowledge-Discovery in Text*) pode ser entendida com um processo de extração e obtenção de informações de qualidade diferenciada a partir de textos que se encontram em língua natural (GUPTA; LEHAL, 2009).

Assim como a Mineração de Dados, que obtém informações de banco de dados e/ou outras fontes estruturadas, a Mineração de Texto se alimenta, principalmente, de dados não estruturados ou semi-estruturados. Essa é a principal diferença entre os dois métodos segundo Gupta e Lehal (2009).

Seu grande potencial se deve ao crescimento contínuo da *Internet* e dos mecanismos de busca que tem sido expressivos nos últimos anos.

Tanto o processo de Mineração de Texto quanto os resultados do mesmo, se diferenciam de uma usual busca na rede mundial de computadores. Neste último caso, observa-se que o agente realizador de uma pesquisa, deixa de lado as partes irrelevantes para focalizar no que julga ter maior importância para agregar valores aos resultados. Diferentemente do primeiro caso onde, segundo Gupta e Lehal (2009), “em Mineração de Texto, o objetivo é a descoberta de informações desconhecidas, algo que ninguém ainda sabe e por isso não poderia estar escrito”.

Após a obtenção do *corpus* (conjunto de documentos recuperados), geralmente, realiza-se alguns processos de pré-processamentos, indexação, mineração *strictus senso* e análise dos resultados. A esse conjunto ordenado de processos dá-se o nome de *Knowledge Discovery in Textual Database (KDT)* que será explicado, com detalhes, na próxima seção.

2.1 *Knowledge Discovery in Textual Database (KDT)*

Apesar da falta de consenso na literatura atual, o *Knowledge Discovery in Textual Database (KDT)*, assim como o *Knowledge Discovery in Database (KDD)*, é um processo (metódico) que tem como intenção a obtenção de conhecimento e pode ser dividido em algumas fases.

Gupta e Lehal (2009) propõem uma estrutura dividida em cinco fases, a saber:

1. Coleta dos dados.
2. Pré-processamento.
3. Mineração de Texto.
4. Gerenciamento de informação.
5. Conhecimento.

Já Aranha *et al.* (2007), Carrilho (2007) propõem um modelo também dividido em cinco fases, porém um pouco diferenciado - modelo este que será utilizado como base para a metodologia de desenvolvimento do presente trabalho. Para estes autores, as fases do *KDT*, são:

1. Coleta.
2. Pré-processamento.
3. Indexação.
4. Mineração de Texto.

5. Análise.

Para esta metodologia faz-se necessário o termino total de uma etapa para avançar para a próxima. É permitida, todavia, a retro-alimentação. Dessa forma, para fins de refinamento, é possível retornar ao processo anterior. O encadeamento de tais fases pode ser apreciado na Figura 1.

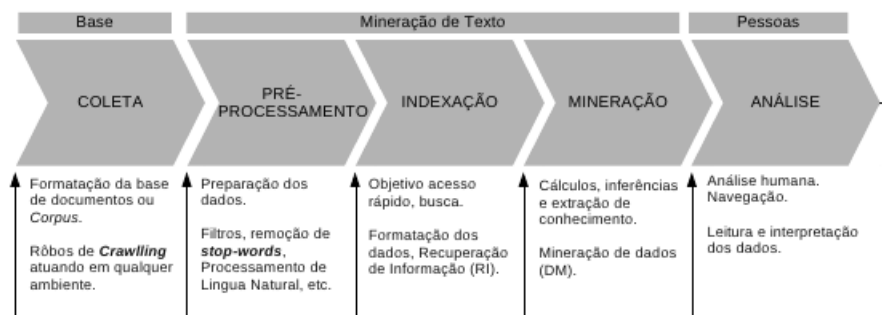


Figura 1: Processo de *KDT* adaptado de Aranha et al (2007)

No decorrer deste trabalho, estas fases serão melhor detalhadas, principalmente nos capítulos referentes à metodologia de desenvolvimento.

Afim de melhor contextualizar o leitor com os aspectos relativos ao processo de *KDT* serão apresentados três seções: uma sobre Recuperação de Informação (2.2), outra sobre Indexação (2.3) e outra sobre Processamento de Língua Natural (2.4).

2.2 Recuperação de Informação

Recuperação de Informação (*RI*) tem como objetivo a automação do processo de recuperação de informações armazenadas em documentos. É uma ciência que pesquisa sobre busca por informações em documentos, busca pelos documentos propriamente ditos, busca por meta-dados que descrevam documentos e busca em banco de dados, sejam eles relacionais e isolados ou banco de dados interligados em rede de hipermídia, tais como a rede mundial de computadores.

Seu principal objetivo é recuperar informação que seja útil para o usuário. Tal informação é normalmente chamada de necessidade de informação do usuário, que não é todavia, uma tarefa fácil de ser caracterizada e executada. Nesse sentido, um sistema de *RI* deve recuperar o maior número possível de documentos relevantes e o menor número possível de documentos irrelevantes.

Como resultado, a presença de apenas, documentos (textos) relevantes entre os documentos retornados por uma consulta, não é uma tarefa simples de ser realizada de forma satisfatória. Uma forma simples de obter um conjunto de respostas para uma consulta de usuário é determinar quais documentos em uma coleção contém as palavras da consulta. Todavia, isto não é o suficiente para satisfazer ao usuário em um sistema de *RI*.

Afim de facilitar o processo de recuperação de informação, os sistemas adotam um vocabulário padronizado, chamado *Vocabulário Controlado*, que pode utilizar uma linguagem natural ou artificial para representar o conteúdo dos documentos. A linguagem natural utiliza os mesmos termos usados pelo autor do documento, enquanto a linguagem artificial adota termos determinados pelos desenvolvedores de tal sistema.

Para uma maior satisfatoriedade devem se levar em conta os conceitos de palavra-chave e grau de relevância, que são componentes cruciais para o cálculo de classificação e ordenação de documentos em um conjunto de respostas a uma consulta.

Para calcular uma classificação é adotado um modelo para representar os documentos e as consultas. Muitos deles tem sido propostos, inclusive com larga adoção de métodos probabilísticos. Os modelos clássicos são:

- Modelo booleano.
- Modelo vetorial.
- Modelo probabilístico.

As principais operações que devem ser implementadas para obtermos uma recuperação satisfatória de informações podem ser resumidos em:

- Operação de Consulta - envolve a especificação de um conjunto de termos, associados ou não por operadores booleanos, que representa a necessidade de informação do usuário.
- Operação de Indexação - envolve a criação de estruturas de dados associados aos documentos de uma coleção. Uma estrutura de dados bastante utilizada são as listas invertidas de termos/documentos.
- Pesquisa e Ordenação - envolve o processo de recuperação de documentos de acordo com a consulta do usuário e sua ordenação através de um grau de similaridade entre o documento e a consulta.

Após executada a tarefa de *RI*, faz-se necessário a execução do pré-processamento e da indexação. Esta última será melhor descrita em termos de referencial teórico na próxima seção e, em termos práticos, na seção 5.4.

2.3 Indexação

O principal objetivo da Indexação é tornar bem definida a estruturação dos dados. Dessa forma, recuperar e manipular os dados se torna uma tarefa mais fácil e menos custosa. Visto que em algumas aplicações a utilização dos dados é feita de forma intensa, o processo de Indexação é fortemente recomendado.

O modelo que mais se destaca e que será utilizado neste trabalho, é o Modelo de Espaço Vetorial (*Vector Space Model*) proposto por Salton (1975). Para este modelo, cada *token* (palavra) é representado como um ponto em um espaço euclidiano $t - dimensional$. Dessa forma, D_i representa o i -ésimo elemento de um documento D com um peso associado.

O resultado da aplicação do Modelo de Espaço Vetorial sobre um documento textual é uma matriz esparsa de grande dimensionalidade (cada dimensão representa um *token*). Conforme demonstrado por Carrilho (2007), uma quantidade de documentos na casa das centenas, pode gerar uma quantidade de *tokens* da ordem de milhares.

Na Tabela 1 é demonstrado em alto nível, como é feita a representação da frase: “Você sabe que não temos tempo, nós não deveríamos tentar. Você sabe que não temos tempo”¹ através do Modelo de Espaço Vetorial. Note que pontuação, espaços e outros caracteres especiais são ignorados.

¹Tradução de “*You know we had no time, we could not even try. You know we had no time*”. Música *Waiting for Darkness* de *Ozzy Osbourne*.

<i>Token</i>	Frequência
você	2
sabe	2
que	2
não	3
temos	2
tempo	2
nós	1
deveríamos	1
tentar	1

Tabela 1: Exemplo de representação do Modelo de Espaço Vetorial.

Em conjunto com o Modelo de Espaço Vetorial, na implementação deste trabalho, utilizou-se do método *term frequency*. Que, basicamente, é um vetor de características com *tokens* ponderados baseando-se na frequência do *token* conforme o exemplo da Tabela 1.

Porém em determinado assunto, alguns termos podem aparecer constantemente - sem que necessariamente sejam relevantes para a extração de opinião. Dessa forma, faz-se necessário um mecanismo capaz de atenuar a influência de tais termos. Para este trabalho, utilizou-se do método de frequência inversa do documento (*inverse document frequency*).

Na Equação 1 temos a Fórmula para frequência inversa de um termo t , onde df_t é a frequência do documento d e t é o termo em questão, nos N documentos do nosso *corpus*:

$$idf_t = \log \frac{N}{df_t} \quad (1)$$

Dessa forma, pode-se demonstrar, através da Equação 2, que a relação $Tf - idf_{t,d}$ (Term frequency-inverse term frequency) retornará o peso de um determi-

nado termo t em um documento d , onde $Tf_{t,d}$ é a frequência do termo t que é multiplicado por $Idf_{t,d}$ que é a frequência inversa dos documentos d em que o termo aparece.

$$Tf - idf_{t,d} = Tf_{t,d} \times Idf_{t,d} \quad (2)$$

Para os autores Manning *et al.* (2008), a relação $Tf - idf_{t,d}$ atribui para um termo t presente em um documento d , um valor:

- Alto, para um termo que ocorre várias vezes em um pequena quantidade de documentos. De forma que este termo tem um alto poder discriminante.
- Baixo, para um termo que ocorre pouco em um documento ou ocorre constantemente em vários documentos. Sua relevância é reduzida.
- Baixíssimo, para um termo que ocorre em praticamente em todos os documentos. Sendo irrelevante, no caso deste trabalho, para a extração de opiniões.

Segundo Carrilho (2007) esta métrica de ponderação dos termos é importante para compor o vetor característica dos documentos, a fim de que se possa calcular com maior precisão Medidas de Similaridade, seja por distância euclidiana, distância baseada em bônus, similaridade por cosseno, etc.

2.4 Processamento de Língua Natural

O Processamento de Língua Natural, doravante *PLN*, é considerado uma subárea da inteligência artificial e da linguística, que estuda e propõe soluções para

os problemas de interpretação e geração automática de línguas humanas naturais. O grande impulso para o surgimento da mesma, no início dos anos 50, foi a tradução automática com finalidade bélica e estratégica feita, inicialmente, de forma rústica tal qual era feito com a criptografia. Apesar de toda essa atenção inicial, houve uma queda de investimentos devido ao desinteresse estratégico e às frustrações iniciais. A busca de formalização de fatores pragmáticos e discursivos, assim como a inserção de gramáticas e analisadores sintáticos só entraram em cena a partir da década de 70 quando essa linha se revigorou de forma genuína.

A arquitetura para um sistema computacional que processa língua natural pode variar dependendo da heurística da problemática envolvida, assim como, a natureza de sua aplicação. Um exemplo completo e complexo, segundo Pria (2008), é um tradutor automático que do ponto de vista de suas funções deverá ser capaz de:

1. Extrair as palavras de uma sentença origem.
2. Analisar sintaticamente esta sentença.
3. Utilizar uma representação intermediária para agregar valor às informações levantadas anteriormente.
4. Analisar semanticamente a sentença afim de obter-se um significado global da mesma.
5. Mapear estes significados extraídos em uma representação adequada.
6. Transformar a representação supracitada em uma sentença destino.

Teoricamente, o processamento de língua natural está intimamente ligado à interação homem-máquina e às técnicas de recuperação de informação.

2.5 Mineração de Texto, Processamento de Língua Natural e aplicações

Como já citado, estimativas apontam que 85% das informações comerciais estão sobre a forma textual (TEXTMININGNEWS, 2011). Além de instigar pesquisadores, empresários e investidores o paradigma tradicional da computação cria grande entraves à obtenção de informação que está relacionada com a dificuldade de captar as relações difusas e, muitas vezes, ambíguas, presentes em documentos de texto. Nesse ponto a Mineração de Texto precisa de métodos e ferramentas com capacidade de captar as informações por trás do emaranhado de palavras, frases e orações dispersas em estruturas (complexas) de linguagem natural. Existe uma relação linguística, psicológica e até mesmo social envolvida nesse cenário incerto, que acima de tudo deve ser tratado por ferramentas que sejam capazes de lidar com imprecisões.

Para que isso possa acontecer necessita-se do apoio do Processamento de Língua Natural, para que se possa, além de realizar a análise do léxico em questão, analisar sintaticamente os textos para definir se estão em uma estrutura base da língua e de qual se trata e semanticamente para verificar a corretude e captar o sentido das sentenças.

Além de todo este apanhado de conceitos e técnicas multidisciplinares, existe uma necessidade por parte dos seres humanos de realizar inferências quanto ao sentimento ou opiniões de outrem. A Análise de Sentimentos ou Mineração de Opiniões, além de caracterizarem a mescla de Processamento de Língua Natural e Mineração de Texto, é responsável por externalizarem tais inferências.

Como exemplos de trabalhos que abordaram as áreas de estudo acima mencionadas, podem ser citadas as aplicações:

- *Wolfram Alpha*: um sistema de buscas que se utiliza de *PLN* para responder perguntas diretas dos usuários (WOLFRAMALPHA, 2012).
- *OpSys*: sistema de mineração de opiniões *web*. Sua versão 2.0 conta com um módulo de análise de investimentos (OPSYS, 2012).
- *Sentweet*: ferramenta especializada em classificar opiniões postadas no *Twitter* (SENTWEET, 2012);

Além de inúmeras aplicações, envolvendo principalmente a mineração de opiniões, existem diversas linhas de pesquisa acadêmica preocupadas em estudar e trazer contribuições positivas para Análise de Sentimento e outras subáreas abordadas neste presente documento. Um exemplo de pesquisa e desenvolvimento acadêmico correlatos é o projeto Brasileiro 2012 desenvolvido no Laboratório de Inteligência Computacional e Sistemas Avançados (*LICESA*) do Departamento de Ciência da Computação da Universidade Federal de Lavras (*UFLA*) (*ESMIN et al.*, 2011).

3 Análise de Sentimento ou Mineração de Opinião

A linha entre a definição das palavras “sentimento” e “opinião” é tênue! Opinião implica em uma conclusão pensada mas que ainda está aberta a discussão. Sentimento sugere uma opinião estabelecida através de sensações de um indivíduo. Neste trabalho será considerado que Análise de Sentimento e Mineração de Opinião possuem conotação idêntica.

A Análise de Sentimento ou Mineração de Opinião, refere-se à extração de informação subjetiva e pessoal através de Processamento de Língua Natural, Lin-

guística Computacional e Mineração de Texto. Como exemplos, temos as situações em que se deseja avaliar as opiniões acerca de um produto ou serviço, e até mesmo, sobre a imagem de alguma empresa, personalidade política ou pública.

Os consumidores, em geral, estão sempre em busca de novas opiniões. Opiniões estas, que tem se multiplicado pela rede mundial de computadores, principalmente com a expansão da *Internet* nas duas últimas décadas (COFFMAN; ODLYZKO, 1998). Segundo Pang e Lee (2008), a maior parte das pessoas asseguram que as informações e opiniões obtidas através de consultas *online* foram determinantes na decisão sobre a compra de algum produto.

De modo geral, Análise de Sentimento tem como objetivo determinar a atitude de um locutor ou um escritor com relação a algum tema ou contexto de um documento. A ação pode ser seu julgamento ou avaliação, estado afetivo (isto é, o estado emocional do autor ao escrever), ou a intenção de comunicação emocional (ou seja, o efeito emocional do autor deseja ter sobre o leitor).

A realização de Mineração de Opinião é comumente feita através da Mineração de Textos, por se tratar de um ramo da mesma, usando elementos e métodos como (maior detalhamento nas subseções seguintes):

1. Máquinas de Aprendizagem
2. Análise de Semântica Latente
3. *Bag of Words*
4. *Support Vector Machines (SVM)*
5. *K-Nearest Neighbours (KNN)*
6. *Multilayer Perceptron (MLP)*

7. Orientação Semântica

Alguns métodos mais sofisticados tentam encontrar o agente (pessoa que mantém um estado afetivo) e o alvo (pessoa ou entidade ao qual o sentimento se refere). Porém a construção prática de sistemas capazes de realizar Análise de Sentimentos de maneira correta – no que tange níveis elevados de taxas de acerto – e em tempo hábil não é uma tarefa das mais corriqueiras ou trivial. Estando, portanto, várias pesquisas sobre o tema em andamento.

Para o presente trabalho, serão utilizados os métodos *Support Vector Machines (SVM)*, *K-Nearest Neighbours (KNN)*, *Multilayer Perceptron (MLP)*. Mais detalhes no Capítulo 5 de metodologia de desenvolvimento.

3.1 Máquinas de aprendizado (*Learning Machines*)

Máquina de aprendizado é um subcampo da Inteligência Artificial e segundo Mitchell (2006) pode ser compreendida pela intersecção da Ciência da Computação com Estatística. Se dedicam, à elaboração de algoritmos e sistemas computacionais capazes de aprender e evoluir seu desempenho em determinadas tarefas com o tempo. Em específico, no presente trabalho, tais tarefas consistem em detecções de padrões utilizados para realizar regressões e predições. O aprendizado pode ser feito de forma supervisionada e não-supervisionada.

O aprendizado realizado pelas Máquinas de Aprendizado é indutivo, ao contrário de outras técnicas de inteligência Artificial que se preocupam com o aspecto dedutivo (Se as premissas são verdadeiras, a conclusão é necessariamente, verdadeira). Em um argumento indutivo, excetuando os argumentos matemáticos indutivos, se a premissa são verdadeiras a conclusão é provavelmente verdadeira, mas

não necessariamente. De forma que argumentos indutivos preservam falsidade, e portanto, técnicas que trabalham sobre aprendizagem indutiva (como é o caso da Aprendizagem de Máquinas) terão que levar este fato em conta.

3.1.1 *Support Vector Machine*

Support Vector Machine (SVM), também é um método que se baseia em técnicas estatísticas. Sendo, pois, do grupo das Máquinas de Aprendizado Supervisionadas para analisar e reconhecer padrões úteis para realizar classificação.

A análise de regressão inclui todas as técnicas para modelagem e análise de diversas variáveis, quando o foco é sobre a relação entre uma variável dependente e uma ou mais variáveis independentes. Útil para se entender a variação de uma variável dependente quando se varia uma variável independente e mantém as outras (independentes) fixas.

Um *SVM* padrão utiliza um conjunto de dados de entrada e prevê, para cada elemento desse conjunto, qual de duas classes o compreende melhor. Caracterizando, assim, o *SVM* como um classificador binário não-probabilístico linear.

Zaghloul *et al.* (2009) demonstraram que este método tem ótimo desempenho para classificação textual, sendo inclusive, um dos mais utilizados para este fim.

Dado um conjunto de exemplos de treinamento, estando os elementos já marcados como sendo pertencentes a uma das duas categorias (classes), um algoritmo de treinamento *SVM* constrói um modelo que atribui novos exemplos a uma categoria ou outra. Este modelo é uma representação dos exemplos como pontos no espaço, mapeados de forma que os exemplos das categorias separadas são divididos por uma lacuna clara que é a mais ampla possível. Novos exemplos são

então mapeados neste mesmo espaço e caracterizados como pertencentes a uma categoria, ou outra, baseando em qual lado da lacuna ele se encontra. Esta lacuna é denominada formalmente de hiperplano.

De maneira formal, um *SVM* constrói um hiperplano (ou um conjunto dos mesmos) em um grande ou infinito espaço t – *dimensional*, que pode ser usado para classificação. Pode-se imaginar que uma boa separação é que possui maior distância (em relação ao hiperplano) entre os pontos mais próximos ao hiperplano. Estes pontos são denominado elementos de borda. Pode-se, finalmente, caracterizar os *support-vectors* como sendo essas distâncias entre os elementos de bordas e o hiperplano, que devem ser maximizados na etapa de treinamento para obtenção de um melhor modelo representacional e conseqüentemente, um melhor resultado. Como exemplo, tem-se a Figura 2 onde $H3$ não é capaz de separar corretamente as duas classes; $H1$ o faz, porém com menor margem; já $H2$ além de separar as duas classes obtém maior margem.

3.2 Análise de Semântica Latente

Análise de Semântica Latente ou *Latent Semantic Analysis (LSA)* é uma técnica da *PLN*, mais precisamente de semânticas vetoriais (*vectorial semantics*) usada para análise, através da Mineração de Texto, da relação entre um conjunto de documentos e os termos contidos nos mesmos. O resultado é um outro conjunto contendo conceitos de relações sobre os documentos e termos.

A *LSA* parte do pressuposto de que palavras com significados semelhantes ocorrerão juntas no texto. Então, uma matriz contendo contagens de palavras por parágrafo (linhas representam palavras únicas e colunas representam cada parágrafo) é construído a partir de um grande pedaço de texto e uma técnica matemá-

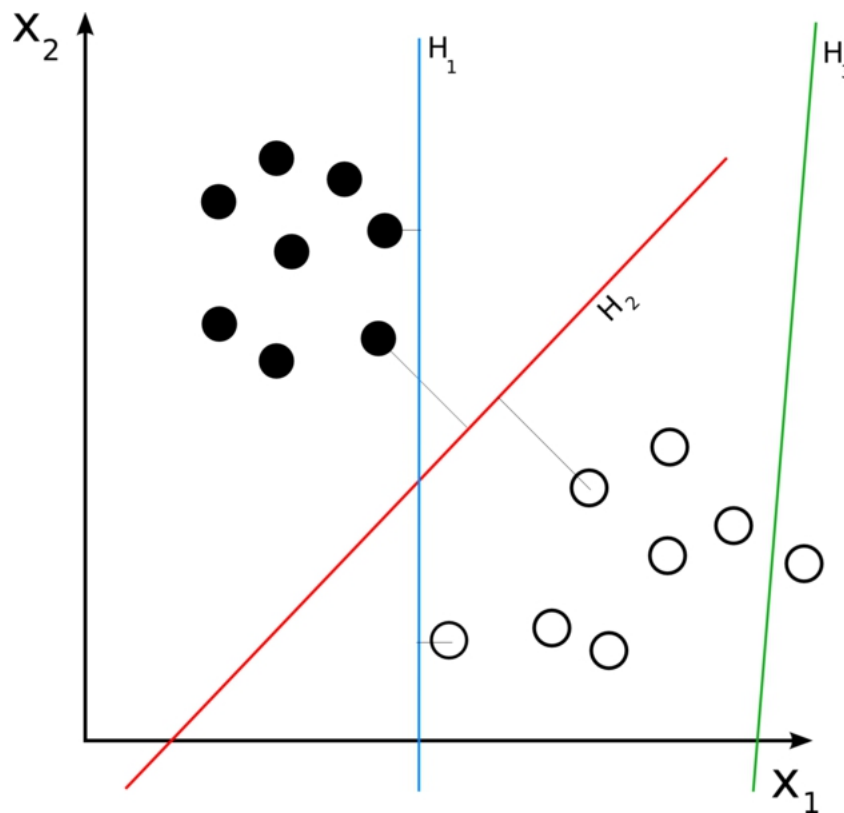


Figura 2: Exemplo de classificação usando SVM

tica chamada *singular value decomposition (SVD)* é usada para reduzir o número de colunas, preservando a estrutura de similaridade entre linhas. As palavras são então comparadas tomando o cosseno do ângulo entre os dois vetores formado por quaisquer duas linhas. Valores próximos a 1 representam palavras muito semelhantes, enquanto valores próximos de 0 representam palavras muito diferentes.

Tomando-se o somatório de duas linhas A e B : $\sum A_i$ e $\sum B_i$, temos que o cosseno de similaridade pode ser calculado pela Fórmula representada na Fórmula 3 e representa a similaridade entre dois vetores.

$$\text{similaridade} = \cos \theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}} \quad (3)$$

3.3 Dificuldades

Para Vigotski (2001), as emoções são históricas e complexas. Seres humanos, enquanto seres dotados de emoções, carregam um inventário imenso de emoções das mais variadas. Emoções estas, que são expressas a todo tempo e em todo lugar.

A palavra, como unidade mínima lexical, deixa de ser apenas um signo (quando dotada de significado e significante) e passa a ser um repositório de idiossincrasias por si própria. “O significado da palavra é microcosmo da consciência humana. A palavra é o signo por excelência” (VIGOTSKI; LURIA, 1994). A palavra, possui um sentido, um dizer, uma significação. E seu potencial para expressar emoções é intangível.

Então o que é emoção? Esta é uma grande questão que vem gerando discussão entre psicólogos e filósofos ao longo de mais de um século, sem que um consenso seja alcançado. Em seu sentido mais literal, o *Oxford English Dictionary* define emoção como “qualquer agitação ou perturbação da mente, sentimento, paixão; qualquer estado mental veemente ou excitado”. Já Goleman (1996), afirma:

“Eu entendo que emoção se refere a um sentimento e seus pensamentos distintos, estados psicológicos e biológicos, e a uma gama de tendências para agir. Há centenas de emoções, juntamente com suas combinações, variações, mutações e matizes. Na verdade, exis-

tem mais sutilezas de emoções do que as palavras que temos para defini-las.”

Fica evidente, desta forma, a dificuldade em se obter opiniões ou emoções (muitas vezes subentendidas e subjetivas) através de meios computacionais automatizados. Esta tarefa delegada a uma máquina digital, é realizada com certa maestria por nós seres humanos, que possuímos um complexo órgão biológico “talhado” e desenvolvido ao longo de milhares de anos, a saber: o cérebro.

Uma primeira dificuldade prática encontrada se refere à coleta de documentos dotados de carga opinativa ou de cunho emocional a respeito de um determinado produto, por exemplo. Essa barreira pode ser transposta obtendo dados de fóruns especializados, utilizando-se uma ferramenta para a coleta específica de informações, ou de sites de *reviews*. Alguns destes sites, inclusive, já possuem métodos de avaliação e medidores.

Outra questão a ser analisada é se queremos classificar um texto opinativo em duas polaridades opostas de sentimentos ou se desejamos encontrar sua localização em espaço contínuo entre estes dois pontos. Grande parte do trabalho sobre classificação subjetiva se enquadra neste cenário.

Eguchi e Lavrenko (2006), apontam que os rótulos de positividade ou polaridade atribuídos podem ser usados simplesmente para resumir o conteúdo das unidades de um texto opinativo sobre um tema, *e. g.*, positivos ou negativos. Assim como, podem ser aplicados para apresentar uma orientação sobre a polaridade da opinião.

4 Redes Neurais

Por mais que a computação tenha evoluído a passos largos nas últimas décadas, o mais fascinante, e ainda incompreensível processador existente é o cérebro humano. O mesmo é complexo e efetua processamento, naturalmente, paralelo; seu armazenamento é adaptativo; possui controle de processos distribuído e capacidade para efetuar o processamento de aproximadamente 10^{11} a 10^{14} operações através de mais de 10^4 ligações entre elementos processados. Em contrapartida, um computador, a priori, efetua processamento sequencial; seu armazenamento é estático; possui controle de processos centralizado e capacidade para efetuar o processamento de aproximadamente 10^5 à 10^6 operações através de menos de 10 ligações entre elementos processados.

Apesar de não estarem totalmente decifrados o processo de formação de pensamentos e de interação entre os neurônios, o conexionismo (*connectionism*) introduziu a ideia de que o cérebro é um sistema altamente distribuído, e é baseado nesse conceito que sua atividade é estabelecida entre os neurônios (RUMELHART *et al.*, 1991). A quantidade de neurônios que compõem tais conexões, em nosso cérebro, são da ordem de 10^9 . Outro ponto positivo para as redes neurais é sua robustez e tolerância a falhas: a eliminação de neurônios não afeta sua funcionalidade global (RAUBER, 2005).

Pode-se dizer que todas as funções, inclusive o pensamento e movimentos do organismo estão intimamente relacionados ao funcionamento destas pequenas células que compõe nosso sistema nervoso.

Os neurônios conectam-se uns aos outros através de sinapses, e juntos formam uma grande rede, chamada rede neural. As sinapses transmitem estímulos

através de diferentes concentrações dos elementos químicos Na^+ (Sódio) e K^+ (Potássio). Este mesmo processo pode ser estendido por todo o corpo humano, e.g., a transmissão de um estímulo sensorial quando queimamos a ponta do dedo. A rede neural garante uma fabulosa capacidade de processamento e armazenamento de informação.

O sistema nervoso, formado por um conjunto extremamente complexo de neurônios, realiza comunicação através de impulsos. Quando um impulso é recebido, o neurônio processa, e passado um limite de ação, dispara um segundo impulso que produz uma substância neurotransmissora que flui do corpo celular para o axônio (que por sua vez pode, ou não, estar conectado a um dendrito de outra célula). O neurônio que transmite o pulso pode controlar a frequência de pulsos aumentando ou diminuindo a polaridade na membrana pós sináptica.

Os neurônios e suas conexões, segundo Rumelhart *et al.* (1986) são os responsáveis pela determinação do funcionamento, comportamento e do raciocínio do ser humano. Ao contrário das redes neurais artificiais, redes neurais naturais não transmitem sinais negativos, sua ativação é medida pela frequência com que se emitem os pulsos. Frequência esta de pulsos contínuos e positivos, enquanto que as redes naturais não são uniformes (apresentam uniformidade apenas em alguns raros pontos do organismo).

Em redes naturais os pulsos não são síncronos, devido ao fato de não serem contínuos, o que a difere de redes artificiais.

Os neurônios são, basicamente, compostos por:

- Os **dendritos**, que têm por função, receber os estímulos transmitidos pelos outros neurônios;

- O corpo de neurônio, também chamado de **somma**, que é responsável por coletar e combinar informações vindas de outros neurônios;
- **Axônio**, composto de uma fibra tubular que pode alcançar até alguns metros, e é responsável por transmitir os estímulos para outras células.

4.1 Redes Neurais Artificiais

Uma rede neural artificial, doravante *RNA*, pode ser entendida como uma técnica computacional que apresenta um modelo matemático baseado na estrutura neural de organismos inteligentes. Formalmente é definida, por Vellasco (1994), da seguinte forma:

“Uma rede neural é um sistema computacional constituído por unidades conhecidas como neurônios. Os neurônios são elementos processadores interligados, trabalhando em paralelo para desempenhar uma determinada tarefa. Os modelos RNAs constituem uma importante técnica estatística não-linear capaz de resolver uma gama de problemas de grande complexidade. Por isso, são modelos úteis em situações que não é possível definir explicitamente uma lista de regras. Em geral, isso acontece quando o ambiente gerador dos dados muda constantemente. As principais áreas de atuação são para classificação de padrões e previsão.”

4.2 Especificações

Sendo, pois, o neurônio a unidade fundamental de uma *RNA*, é apresentado na Figura 3 um esquema funcional de um *k-ésimo* neurônio em uma rede, *i. e.*, sua estrutura esquemática interior.

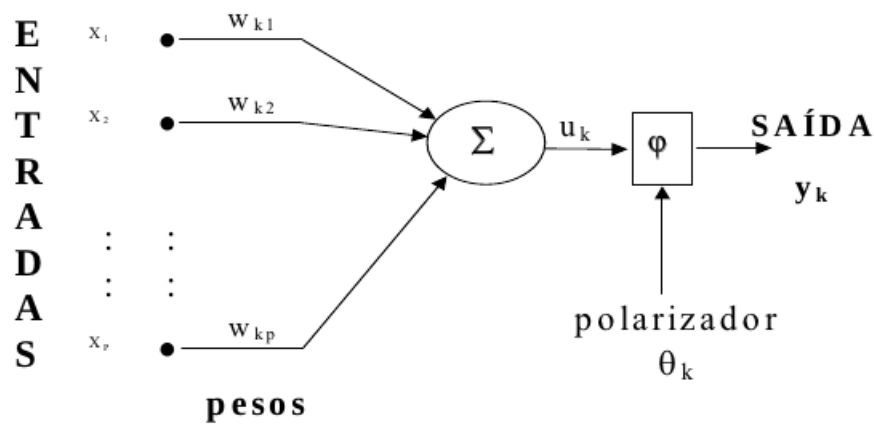


Figura 3: Esquema de um neurônio adaptado de Vellasco, M. M. B. R. (1994)

As entradas estão representadas pelos x 's. Esses x 's não são os padrões da camada de entrada, mas a saída do neurônio da camada anterior.

Os w 's representam os pesos (parâmetros da rede). Os parâmetros representam a "memória" da rede, *i. e.*, a experiência ganha como resultado das n -apresentações dos padrões. São os pesos que combinam a não-lineariedade para que a mesma fique distribuída pela rede.

U_k representa a combinação linear dos pesos. Corresponde a soma ponderada da entrada dos pesos.

Y_k é a saída do *k-ésimo* neurônio que depende do nível de ativação aplicado ao neurônio pela função de ativação. Ressalta-se que a função de ativação refere-se a

parte não-linear de cada neurônio, sendo o único local onde tal não-linearidade se encontra.

θ_k é conhecido como termo polarizador. É apresentado, algumas vezes na literatura como *threshold* e define o domínio dos valores de saída. Na modelagem recorre-se ao artifício de tratar este termo como mais um peso (*bias*) de modo que, durante o processo de otimização dos pesos, a ser realizado pelo algoritmo implementado, a atualização aconteça para todos os parâmetros, incluindo para o polarizador.

Em uma rede neural os parâmetros a serem estimados são os pesos e o polarizador. Como em cada neurônio chega a soma ponderada de todas as entradas, então o polarizador aparecerá associado a uma entrada fixa +1 ou -1 (VELLASCO, 1994).

4.3 *Feedforward Neural Network*

Ao contrário das redes neurais recorrentes, onde se encontram ciclos de instruções na topologia da rede, uma *feedforward neural network* é linear. Nesta rede, a informação se move em uma única direção, para frente, a partir dos nós de entrada, através de nós ocultos (se houver) e para os nós de saída. Não há ciclos ou loops na rede, o que a torna mais simples sendo um dos primeiros tipos de RNAs criadas.

Não possuindo, pois, realimentação da saída para a entrada, tais redes são consideradas “sem memória”. Abaixo segue um exemplo (Figura 4) da topologia de uma *single-hidden layer feedforward networks (SLFNs)*, que se trata de uma RNA não-recorrente contendo apenas uma camada interna. Sobre tal arquitetura é que se pretende aplicar o algoritmo de *ELM* para o propósito deste trabalho.

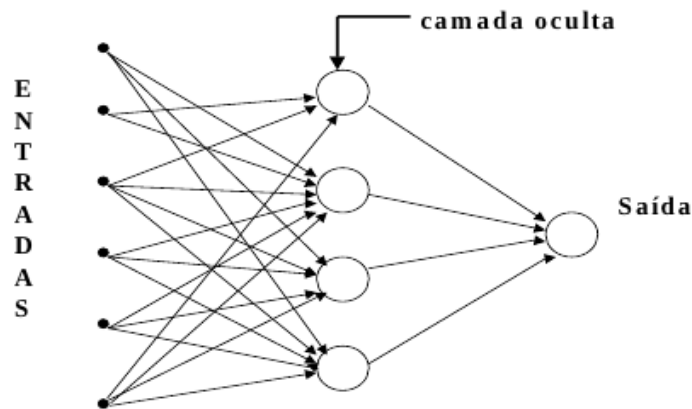


Figura 4: *Single-hidden layer feedforward network* adaptado de Vellasco, M. M. B. R. (1994)

Apesar dos vários tipos de *RNs* existente, *feedforward neural networks* são, provavelmente, as mais populares segundo Huang *et al.* (2011). Deve-se ressaltar que, entre a camada de entrada (que recebe estímulo do ambiente externo) e a camada de saída (que envia a saída para o ambiente externo), podem existir uma (*single-hidden layer*) ou mais (*multi-hidden layers*) camadas escondidas.

Três principais abordagens são comumente usadas para treinar uma *feedforward network*:

1. Baseadas em gradiente descendente, por exemplo, *backpropagation*. O gradiente é a primeira força dos pesos, no caso uma função linear de pesos. O algoritmo de declive descendente é baseado numa correção a qual é proporcional ao gradiente. Assim, a correção é proporcional a uma função linear de pesos. Tal abordagem só é passível de ocorrer em redes com retroalimentação.

2. Baseadas em métodos de otimizações padrões. É o caso do *Support Vector Machines (SVM)* para um tipo de *SLFNs*, chamadas de *Support Vector Network*.
3. Baseadas em técnicas quadrados mínimos. Por exemplo, radial basis function (*RBF*) *network learning* (LOWE, 1989)

4.4 *Extreme Learning Machine*

Extreme Learning Machines (ELM) foram originalmente desenvolvidas para *SLFNs* (HUANG *et al.*, 2004). Sua essência é de que, diferentemente do entendimento de “aprendizagem”, a camada escondida de uma *SLFN* não precisa ser ajustada. A implementação típica de uma *ELM* aplica nós de computação aleatória, que devem ser independentes dos dados de treinamento.

Ao contrário dos métodos tradicionais de aprendizado para redes neurais, *ELM* não só tende a atingir o menor erro de treinamento, mas também a menor norma dos pesos de saída.

Segundo as teorias de Bartlett (1998), para redes neurais do tipo *feedforward* buscando por menores erros de treinamento, a norma dos pesos é a menor generalização que a rede tende a ter. Uma vez que para as *ELMs* os pesos das camadas internas (escondidas) não precisam ser ajustadas e seus parâmetros podem ser fixos, o peso das saídas podem ser resolvidos pelo método dos quadrados mínimos.

Características de uma *ELM*:

1. Sua camada interna não necessita de ser ajustada iterativamente (HUANG *et al.*, 2004; HUANG *et al.*, 2006).

2. De acordo com a teoria, apresentada por Bartlett (1998), sobre *feedforward neural network*, tanto o erro de treinamento $\|H\beta - T\|$ quanto a norma dos pesos $\|\beta\|$ precisam ser minimizados.
3. A camada interna necessita de satisfazer condições mínimas de aproximações, para um bom resultado por parte da *ELM*.

4.4.1 Funcionamento básico

Os parâmetros dos nós (neurônios) da camada interna são gerados aleatoriamente e permanecem fixo. Treinar uma *SLFN* é equivalente a encontrar a matriz unidimensional (solução) β do sistema linear $HT = \beta$, através do método de quadrados mínimos: $\|H\beta - T\| = \min\|H\beta - T\|$

Se o número L de neurônios da camada interna for igual ao número N de diferentes amostras de treinamento, $L = N$, então a matriz H é quadrada e poderá ser invertível quando os parâmetros (a_i, b_i) para tais neurônios forem gerados aleatoriamente. Dessa forma, a resolução do sistema linear é possível e exata de forma que a *SLFN* pode aproximar com erro zero para as referidas amostras. Entretanto, na maioria dos casos, o número de neurônios internos é muito menor do que a quantidade de amostras, distintas, *i. e.*, $L \ll N$. Dessa forma, H deixa de ser uma matriz quadrada e não existem a_i, b_i, β_i tal que $H\beta = T$. A menor norma da solução dos quadrados mínimos para o sistema linear é: $\beta = H^+T$ onde H^+ é a matriz inversa de H . Podemos sumarizar o funcionamento de uma *ELM* através do seguinte algoritmo:

Dados como entrada:

- Um conjunto de treinamento:

$$\varphi = (x_i, t_i) | x_i \in \mathbb{R}^d, t_i \in \mathbb{R}^m, i = 1, \dots, N$$

- Uma função de saída, para os neurônios da camada interna:

$$G(a_i, b_i, x)$$

- Um número L de neurônios nessa camada

Executar os seguintes passos:

1. Calcular a matriz H de saída para a camada interna.
2. Calcular o peso do vetor β :

$$\beta = H^T T$$

O algoritmo para *ELM* funciona para uma grande quantidade de tipos de função de ativação (SIEW *et al.*, 2004). Muitos algoritmos de aprendizado não lidam com redes de limiar (*threshold network*) diretamente, em vez disso, redes análogas são usadas para obter resultados aproximados aos de uma rede de limiar de forma que o método gradient-descent, finalmente, pode ser usado como treinamento.

Em contrapartida, *ELM* pode ser usada diretamente para treinar redes de limiar (HUANG *et al.*, 2006). Diferentes métodos podem ser usados para calcular a matriz inversa de Moore-Penrose:

- Método da projeção ortogonal
- Método da ortogonalização

- Métodos iterativos
- Decomposição de valor singular (*SVD*)

4.4.2 Vantagens

Como uma técnica de aprendizagem, *ELM* tem demonstrado grande potencial para resolver problemas de classificação (que é o alvo de nosso interesse no presente trabalho) e regressão. Por se tratar de uma técnica recente, tem atraído considerável atenção por parte das comunidades de Inteligência Artificial e de Máquinas de Aprendizado.

Algumas questões ainda estão em aberto, por exemplo, prova teórica e matemática do porquê de uma *ELM* ser pouco sensível à quantidade de neurônios da camada interna (comparado ao algoritmo de *Backpropagation*). Outras questões foram provadas e demonstradas experimentalmente por Huang *et al.* (2006), a saber:

1. A velocidade de treinamento de uma *ELM* é extremamente rápida, completando esta fase em segundos ou menos de um segundo e alcançando resultados que chegam a ser 170 vezes mais rápido do que o *backpropagation*, 430 vezes mais rápido do que o *SVM* e 10.000 vezes mais rápido do que o *SRV*.
2. Melhor generalização do que os métodos baseados em gradiente descendente e *backpropagation*, na maioria dos casos.
3. Tradicionalmente, todos os parâmetros de uma rede unidirecional precisam ser ajustados iterativamente. Em uma *ELM*, tais parâmetros são gerados randomicamente e os pesos atribuídos analiticamente.

4. Métodos clássicos de aprendizagem baseado em gradientes podem enfrentar vários problemas como: mínimos locais, taxa de aprendizagem inadequada (rede divergente), *overfitting* (super-treinamento, *i. e.*, generalização prejudicada), etc.

5 Metodologia de desenvolvimento

O intuito do trabalho realizado, consiste na implementação de um algoritmo *ELM* a ser comparado com o *SVM*², *KNN* e *Multilayer Perceptron*, já existentes na suíte comumente usada *Weka*. Além de conter algoritmos de pré-processamentos, indexação, normalização, entre outros que serão muito úteis, o *Weka* contém dezenas dos principais algoritmos para mineração de dados (HALL *et al.*, 2009). Implementado na linguagem *Java* e passível de inserir novos algoritmos (na mesma linguagem) na ferramenta, o *Waikato Environment for Knowledge Analysis (Weka)* obtém grande visibilidade dentre o meio acadêmico e até empresarial, por ter sido desenvolvido na Universidade de Waikato na Nova Zelândia, pela sua qualidade e documentação.

Antes de entrarmos nos detalhes sobre a metodologia de desenvolvimento é necessário conhecer como as mensagens foram classificadas e as métricas utilizadas.

5.1 Classificação das mensagens

A título de melhor entendimento do problema, serão apresentados alguns exemplos de mensagens e suas classificações na Tabela 2. Erros de ortografia, pontuação, *smileys* entre outros, foram mantidos para exemplificar uma situação real.

É importante observar que **todos** os experimentos foram realizados através da técnica de *K-folds cross-validation*. No *K-fold cross-validation*, a amostra original é aleatoriamente dividida em *k* subamostras. Dessas *k* subamostras, uma

²Importado para o *Weka* conforme especificado na seção 5.5.1.

Mensagem	Classe
Bahia é, com certeza, o pior time do brasileiro 2012.	Negativo
Ta bom né, meu colorado venceu :D Otima semana já!	Positivo
No corpo, alma e mente, sou Palmeiras eternamente..	Positivo
Pior que o Internacional nãoo tem. É noiss Grêmio!!!	Conflitante
De volta ao Palmeiras, Tadeu treina e aguarda proposta de outros clubes	Indefinido

Tabela 2: Exemplo de mensagens e suas classes

subamostra única é mantida como dado de validação para testar o modelo, e as $k - 1$ subamostras restantes são usadas como dados de treinamento. O processo de validação cruzada é então repetido k vezes (os *folds*), com cada uma das k subamostras utilizadas exatamente uma vez como dado de validação. Os K resultados, em seguida, são usados para gerar uma média (ou outra métrica) para produzir uma estimativa única. A vantagem deste método ser repetido sobre subamostras aleatórias, é que todas as subamostras são utilizadas tanto para treinamento e validação e cada qual é usada para a validação exatamente uma vez. O método 10-fold cross-validation é comumente utilizado, mas em geral k continua a ser um parâmetro.

Para a realização experimental dos algoritmos, utilizou-se como base as seguintes métricas: Precisão, *Recall*, *F-measure*, *Estatística Kappa*, Raiz do Erro Quadrático Médio e Desvio Padrão.

5.2 Métricas

A seguir, tem-se uma breve explicação (e posteriormente as definições matemáticas) das métricas utilizadas no presente trabalho:

- Percentual de acerto: porcentagem de acerto do algoritmo quando utilizado nas bases de testes.

- Tempo de treinamento: tempo que o algoritmo requer para gerar seu modelo de classificação.
- Precisão (π): representa o quanto a classificação feita por um algoritmo corresponde à realidade (classificação manual feita por um especialista).
- *Recall* (ρ): mede o quanto a classificação feita por um especialista coincide com a realizada pelo algoritmo.
- *F-measure* (F_β): representa uma relação de correspondência entre Precisão e *Recall*.
- *Estatística Kappa* (κ): indica o grau de concordância entre dois classificadores, levando em consideração a probabilidade de as concordâncias terem acontecidas ao acaso.
- Raiz do Erro Quadrático Médio (*RMSE*): raiz quadrada do erro quadrático médio.
- Desvio Padrão (σ): medida de dispersão dos valores de uma distribuição normal em relação à sua média.

Ambos, Precisão (π) e *Recall* (ρ), são definidos, segundo Lin *et al.* (2009) a partir de probabilidades condicionais através de uma Tabela de contingência, para determinada mensagem i e classe C_i .

	Classificação do especialista		
	Sim	Não	
Classificação do algoritmo	Sim	VP_i	FP_i
	Não	FN_i	VN_i

Tabela 3: Tabela de contingência. Adaptado de Lin, Hsieh e Chuang 2009.

Na Tabela 3 temos que FP_i (Falso Positivo) é o número incorreto de mensagens classificadas como positivas. VP_i (Verdadeiro Positivo) é o número de mensagens positivas classificadas corretamente. FN_i (Falso Negativo) é o número incorreto de mensagens classificadas como negativas. VN_i (Verdadeiro Negativo) é o número de mensagens negativas classificadas corretamente.

Tendo isso em mente, podemos calcular Precisão (π) e *Recall* (ρ) e *F-measure* (F_β), respectivamente, através das seguintes equações:

$$\pi_i = \frac{VP_i}{VP_i + FN_i} \quad (4)$$

$$\rho_i = \frac{VP_i}{VP_i + FP_i} \quad (5)$$

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho} \quad (6)$$

Sendo β (Equação 6) o grau de importância dos atributos π e ρ , temos, por dedução, que se $\beta = 0$ então F_β coincide com π ; por outro lado, se $\beta = +\infty$ então F_β coincide com ρ .

Com o intuito de normalizar os resultados, por definição, no presente trabalho $\beta = 1$ de forma que, o resultado para *F-measure* está compreendido no seguinte intervalo $0 < F_\beta < 1$. Assim, quando o resultado de F_β estiver próximo de zero (0) o algoritmo estará com uma taxa grande de erro em relação à classificação feita pelo especialista; quando F_β estiver próximo de um (1) o algoritmo estará com uma taxa grande de acerto em relação à classificação feita pelo especialista (LIN *et al.*, 2009).

A *Estatística Kappa*, segundo Landis e Koch (1977) indica o grau de concordância entre dois classificadores, levando em consideração a probabilidade de as concordâncias terem acontecidas ao acaso. Para isso é introduzido o conceito de **Classificador Infalível**, o qual é utilizado para confrontar os dados com o classificador real levando em consideração as escolhas aleatórias.

O valor de *Kappa* está compreendido no intervalo $0 \leq \kappa \leq 1$ e pode ser interpretado segundo a Tabela 4.

κ	Interpretação
0	Nenhuma concordância
0 a 0,2	Leve concordância
0,21 a 0,4	Concordância regular
0,41 a 0,6	Concordância moderada
0,61 a 0,8	Concordância substancial
0,81 a 1,0	Concordância quase perfeita

Tabela 4: Interpretação dos valores de *Estatística Kappa*. Extraído de Landis e Koch 1977

A Raiz do Erro Quadrático Médio (*RMSE*), uma medida estatística bastante conhecida e utilizada, pode ser definida segundo a Equação 7, onde d_i é o valor previsto, y_i é o valor obtido e n é a quantidade de amostras.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (d_i - y_i)^2}{n}} \quad (7)$$

O Desvio Padrão σ , é outra medida estatística clássica, que pode ser entendida como uma medida de dispersão dos valores de uma distribuição normal em relação à sua média. Podemos observar na Equação 8 que o Desvio Padrão σ é a raiz quadrada da variância dos dados x_i em relação à média \bar{x} para uma medida amostral de n elementos.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}} \quad (8)$$

Todas essas métricas, são de grande valia para o entendimento dos resultados e análise dos mesmos que será realizada no Capítulo 6.

5.3 Coleta dos dados

Para a coleta dos dados necessários à realização do preterido trabalho, utilizou-se do serviço de *micro blogging* lançado em 2006, a saber: *Twitter*³. Um dos grandes motivacionais para a escolha foram: o seu grande número de usuários – estimados em 175 milhões pelo próprio site em setembro de 2010 – e obviamente, grande número de informações.

Uma de suas características é de que o serviço permite postagem de mensagens de tamanho máximo prefixado em 140 caracteres. Dessa forma, temos como ponto positivo um bom limite superior (em termos de caracteres) para as mensagens. Como ponto negativo para a classificação das mensagens temos a linguagem informal, irônica e despojada dos usuários.

Para a coleta das mensagens, utilizou-se de uma aplicação desenvolvida anteriormente pelo autor deste trabalho em um projeto do Laboratório de Inteligência Computacional e Sistemas Avançados da Universidade Federal de Lavras (ESMIN *et al.*, 2011). A saber, uma ferramenta simples de coleta, armazenamento e classificação de mensagens do *Twitter* implementada em *AJAX (PHP⁴, JavaScript, JQuery, JSON, MySQL)*.

³<http://www.twitter.com>

⁴<http://ww.php.net>

Coletou-se uma quantidade total de 1000 mensagens para serem analisadas e classificadas em duas polaridades opostas de sentimentos. Não optou-se por encontrar sua localização em espaço contínuo entre estes dois pontos. Grande parte do trabalho sobre classificação subjetiva se enquadra neste cenário. Os autores Eguchi e Lavrenko (2006), apontam que os rótulos de positividade ou polaridade atribuídos podem ser usados simplesmente para resumir o conteúdo das unidades de um texto opinativo sobre um tema, por exemplo, positivos ou negativos. Assim como, podem ser aplicados para apresentar uma orientação sobre a polaridade da opinião.

Como estratégia para a realização dos experimentos, dividiu-se os dados coletados em:

- Dados classificados manualmente, por um especialista, para treinamento dos algoritmos. Sua parcela é de 80% (800 mensagens) dos dados totais.
- Dados não-classificados para execução dos experimentos. Sua parcela é de 20% (200 mensagens) dos dados totais.

A Tabela 5 permite uma melhor visualização das distribuições dos dados.

Classe	Quantidade	Percentual (total)
Positivas	400	40%
Negativas	400	40%
Não classificadas	200	20%

Tabela 5: Distribuição dos dados

Vale frisar que os **mesmos** dados, tanto os classificados manualmente como os não-classificados, foram utilizados para o treinamento e execução, respectivamente, de todos algoritmos de forma a não permitir tendenciosidades.

5.3.1 Método de coleta

A coleta de dados do serviço de *micro blogging* supracitado pode ser realizado, sem muitos entraves, através da *API (Application Programming Interface)* de desenvolvimento oficial do próprio *Twitter*.

Sendo, pois, inteiramente baseadas em métodos *HTTP (Hypertext Transfer Protocol)*, requisições podem, facilmente, ser realizadas através da *API* pelo método *GET*.

Para a automação do processo, foi criado um módulo no sistema de coleta e classificação. Constitui-se basicamente de um *script* na linguagem *PHP* responsável por realizar as requisições através da referida *API* e armazenar os resultados em um banco de dados relacional. O banco adotado foi o *MySQL* por ser gratuito, rápido e, segundo os próprios mantenedores do banco de dados, o mais utilizado no mundo.

5.3.2 Critérios adotados para coleta

O tema para as mensagens coletadas foi o Campeonato Brasileiro de Futebol, popularmente conhecido como Brasileirão. Optou-se, por este tema, pois é um assunto bastante comentado e já foi trabalhado anteriormente, pelo autor deste documento, em outros projetos.

Dessa forma, selecionou-se uma lista com os 20 times classificados para o Campeonato Brasileiro de Futebol do ano de 2012. Coletou-se uma quantia de 50 *twittes* para cada time, totalizando 1000 mensagens. Os times são: Atlético Goianiense, Atlético Mineiro, Bahia, Botafogo, Corinthians, Coritiba, Cruzeiro, Fi-

gueirense, Flamengo, Fluminense, Grêmio, Internacional, Palmeiras, Santos, São Paulo, Vasco, Náutico, Portuguesa, Ponte Preta e Sport.

5.3.3 Classificação manual das mensagens

Após serem coletadas (com base nos critérios da seção anterior) e armazenadas em banco de dados, as mensagens foram classificadas manualmente, por uma equipe de sete especialistas do *LICESA* (ESMIN *et al.*, 2011). As classes possíveis são:

- Positivo: Mensagens que favorecem o time em questão.
- Indefinido: Mensagens neutras que não trazem aspectos positivos nem negativos para o time em questão.
- Negativo: Mensagens que desfavorecem o time em questão. São divididas em sub-classes: Tristeza, Medo, Raiva, Desgosto e Surpreso. Para este trabalho, em especial, as sub-classes não foram levadas em questão.
- Conflitante: Mensagem que apresenta opinião sobre mais de um time. Estas mensagens foram descartadas para não criar ambiguidade.

Somente as mensagens positivas e negativas foram efetivamente utilizadas neste trabalho. Mensagens indefinidas ou conflitantes foram descartadas pois fogem ao escopo do problema que estamos atacando, conforme definido na introdução do Capítulo 5. Essa medida, previne inclusive, que a qualidade dos classificadores seja afetadas.

5.4 Pré-processamento e indexação

Após a coleta e classificação manual das mensagens, fez-se necessário a realização do pré-processamento das mesmas. A principal intenção é realizar uma “limpeza” nos dados. Primeiramente é realizada a *tokenização* (que consiste em separar palavra por palavra, removendo espaços em branco, fim de linha, tabulações e etc) dos termos e em seguida efetua-se os seguintes passos:

1. Remoção de *StopWords*, que são palavras irrelevantes para aplicação, como: artigos, preposições, numerais.
2. Remoção de caracteres inválidos.
3. Remoção de dígitos.
4. Remoções de *links*.

Essa etapa foi realizada, no ambiente *WEKA*, através do filtro *StringToWordVector*, com a mesma configuração para todos os experimentos. Além de pré-processar os dados (através da definição de expressões regulares), a intenção em utilizar o *StringToWordVector* é de que esse filtro realiza uma conversão dos dados para o Modelo de Espaço Vetorial, através das métricas TF-IDF (já explicadas anteriormente na seção 2.3).

Após essa etapa, os dados encontram-se preparados para a mineração *stricto sensu*. É importante frisar que, a etapa de pré-processamento e indexação foi realizada exatamente da mesma forma para todos os métodos.

5.5 Mineração: Análise de Sentimento

Esta seção é destinada a apresentar a metodologia de desenvolvimento da Mineração de Opinião. Primeiro será discorrido sobre o procedimento e ferramenta utilizada para a realização dos experimentos de mineração através do *SVM*, *KNN*, *MLP* e, em seguida, do *ELM*.

5.5.1 Support Vector Machine (SVM)

Para a execução da mineração através do algoritmo *SVM* utilizou-se da já comentada suíte para mineração de dados *Weka*. Um forte motivacional para a utilização do *Weka* está no fato de que os algoritmos tem sido desenvolvidos e melhorados ao longo de anos por especialistas ao redor do mundo, por se tratar de um projeto *open source*⁵, possuir ampla gama de algoritmo já otimizados e capacidade de inserir novos algoritmos desenvolvidos pelos usuários.

Nativamente o *Weka* não traz uma implementação do *SVM*, porém a inclusão é facilmente realizada a partir do seguinte comando⁶:

```
java -classpath $CLASSPATH:weka.jar:libsvm.jar weka.gui.GUIChooser
```

Para que o comando surta efeito é necessário que o diretório corrente contenha os arquivos *weka.jar* e *libsvm.jar*. Este último arquivo contém a implementação do *SVM* e foi desenvolvido por EL-Manzalawy e Honavar (2005), responsáveis pelo provejo *WLSVM*.

⁵Projetos de códigos abertos livremente disponibilizados para que qualquer pessoa possa utilizar e modificar.

⁶Comando para sistemas operacionais Linux

Na execução do *SVM* é necessário definir alguns parâmetros do algoritmo como: *cacheSize*, *coef0*, *cost*, *degree*, *gamma*, *normalize*, *weights*, etc. Em cada tipo de problema a ser atacado uma variação na configuração desses parâmetros pode reproduzir um resultado melhor ou pior. Para o presente trabalho, todos os parâmetros foram mantidos na configuração *default* (padrão) para problemas de classificação, para não criar-se tendências.

Os resultados da execução do *SVM* estão detalhados no Capítulo de resultados 6, mais especificamente na seção 6.1.

5.5.2 *K-Nearest Neighbours (KNN)*

Para a realização do *K-Nearest Neighbours (KNN)*, utilizou-se do algoritmo *ibk* (da categoria *lazy*) que já vem, por padrão, incluso no *Weka*. Faz-se necessário, configurar alguns parâmetros de entrada para o algoritmo como: *k*, *crossValidade*, *distanceWeighting*, *meanSquared* e etc.

Na execução dos experimentos a partir do *KNN* optou-se por ajustar o parâmetro *k* com valor 1 e *crossValidade* com valor *true*. Isso se deve ao fato, de que, para o presente trabalho, constatou-se empiricamente através do método da tentativa e erro, que para $k=1$ o *KNN* obtém melhor acurácia e desempenho. Já o *crossValidade* foi mantido em *true* para que fosse efetuado o *cross-validation*, conforme explicado anteriormente no início do Capítulo 5. O restante dos outros parâmetros foram mantidos em seus valores *default*.

5.5.3 *Multilayer Perceptron (MLP)*

Para a execução do algoritmo *Multilayer Perceptron (MLP)*, utilizou-se a função *MultilayerPerceptron* (da categoria *functions*) que já vem, por padrão, inclusa no *Weka*. Assim como os outros métodos, deve-se setar alguns parâmetros de entrada para o algoritmo como: *decay*, *hiddenLayers*, *neuronsLayer*, *LearningRate*, *momentum*, entre outros, num total de quinze (15) parâmetros. A maioria deles mantidos em seu valor *default*.

A fim de obter o melhor resultado e desempenho possível, fez-se necessário ajustar os valores de *hiddenLayers*, *neuronsLayer*, *LearningRate* e *momentum* em: 2, 9, 0.4 e 0.2, respectivamente. Dessa forma, a rede neural terá uma configuração de duas camadas, cada uma com 9 neurônios e sua taxa de aprendizado será $\alpha = 0.4$ e momentum $\beta = 0.2$. Tais valores foram encontrados empiricamente através do método da tentativa e erro, a fim de se obter uma boa relação entre acurácia e desempenho.

5.5.4 *Extreme Learning Machine (ELM)*

Não há conhecimento, até o presente momento, de algum algoritmo *Extreme Learning Machine* já implementado para o *Weka*, ou mesmo para a linguagem *Java* - linguagem esta na qual o próprio *Weka* foi desenvolvido. Então, fez-se necessário a implementação (em linguagem *Java*) do referido algoritmo conforme o algoritmo clássico proposto por Huang *et al.* (2004), já descrito na sub-seção 4.4.1. A implementação se deu tal qual as especificações e necessidades do *Weka* para que fosse possível sua importação na suíte. Dessa forma, obtem-se algumas vantagens, como o aproveitamento de algumas rotinas já implementadas na suíte,

principalmente para o cálculo das métricas: Precisão (π) e *Recall* (ρ), *F-measure* (F_β), *Estatística Kappa* (κ), Raiz do Erro Quadrático Médio (*RMSE*) e Desvio Padrão (σ).

Após a implementação do algoritmo, utilizou-se como parâmetros de entrada:

- Arquivo contendo os dados de entrada já pré-processados e indexados.
- Quantidade de neurônios para a camada escondida da *Single-Hidden Layer Feedforward Network (SLFN)*.

Foram realizados alguns testes, para obter obter um melhor domínio sobre o problema. Basicamente, executou-se o algoritmo algumas vezes a fim de encontrar um número adequado de neurônios para a camada escondida da *SLFN*. Este número adequado precisava satisfazer três condições: (1) boa acurácia, (2) uso de memória, (3) tempo de treinamento. Como se trata de uma Máquina de Aprendizado Extremo, a última condição não interferiu, pois, o algoritmo sempre convergia para uma solução em tempo absolutamente mais rápido do que outras Máquinas de Aprendizado comumente usadas (inclusive *SVM*). Essa característica intrínseca e teórica, foi o grande motivacional para a realização do presente trabalho.

6 Resultados e análise

Para uma melhor separação e compreensão dos resultados dos experimentos, cada método – *SVM* (6.1), *KNN* (6.2), *Multilayer Perceptron* (6.3), *ELM* (6.4), – será analisado separadamente nas próximas seções do presente capítulo. Ao final do mesmo, será feito um comparativo do melhor resultado e das médias de cada um dos métodos na Seção 6.5.

Foram realizados, para cada um dos já citados métodos, uma quantidade de dez (10) experimentos usando a técnica de *Cross-validation* (mais precisamente *10-folds cross-validation*) para cada um deles, onde cada experimento foi repetido dez (10) vezes. Dessa forma, a coluna **Experimento** representa a média dessas dez execuções e a linha final da tabela traz a média de todos os dez experimentos. Todos experimentos foram executados em uma mesma máquina de configuração: 4GB de memória RAM e processador AMD Phenom(tm) II X4 B93 com 4 núcleos de processamentos de 2,7GHz cada e 512 KB de cache.

As métricas que serão levadas em conta serão:

- Porcentagem de acerto (%).
- Tempo de treinamento (τ) em segundos.
- Tempo de execução (ε) em segundos.
- Precisão (π).
- *Recall* (ρ).
- *F-measure* (F_{β}).
- *Estatística Kappa* (κ).

- Raiz do Erro Quadrático Médio (*RMSE*).
- Desvio Padrão (σ).

Sempre que uma execução dos experimentos se sobressair, a mesma será destacada em tom cinza nas tabelas. Caso haja mais de uma execução que se sobressaia, elas serão destacadas.

6.1 SVM

Conforme já mencionado na Seção 5.5, para a experimentação, utilizou-se do algoritmo *SVM* desenvolvido pelo projeto *WLSVM* que foi importado para o *Weka*. A seguir, na Tabela 6, encontram-se os resultados obtidos:

Experimento	Acerto(%)	π	ρ	F_β	κ	<i>RMSE</i>	σ
1	70.89	0.75	0.61	0.67	0.43	0.54	0.07
2	69.46	0.73	0.62	0.67	0.39	0.55	0.06
3	71.79	0.76	0.64	0.69	0.44	0.53	0.04
4	69.46	0.72	0.65	0.68	0.39	0.55	0.04
5	68.93	0.72	0.64	0.67	0.38	0.55	0.06
6	73.75	0.79	0.65	0.71	0.48	0.51	0.05
7	68.57	0.70	0.65	0.67	0.37	0.56	0.04
8	73.93	0.77	0.69	0.72	0.48	0.51	0.05
9	72.32	0.76	0.66	0.70	0.45	0.52	0.06
10	74.11	0.76	0.71	0.73	0.48	0.51	0.06
Média	71.32	0.74	0.65	0.69	0.42	0.53	0.05

Tabela 6: Resultados da execução do *SVM*.

Podemos ver que o experimento **10** obteve maior taxa de acerto e terceira maior Precisão (π). De uma forma geral, devido às características do problema, o *SVM* obteve bons resultados, inclusive com pequeno desvio padrão (σ) para todos os experimentos.

A seguir, na Tabela 7, temos o tempo de treinamento do modelo e tempo de execução do mesmo sobre os dados de teste não classificados para o *SVM*.

Experimento	τ (s)	ε (s)
1	16.84	0.01
2	17.06	0.01
3	15.96	0.01
4	16.06	0.01
5	18.07	0.01
6	20.02	0.01
7	15.76	0.01
8	16.05	0.01
9	19.07	0.01
10	21.03	0.01
Média	17.59	0.01

Tabela 7: Tempo de treinamento (τ) e execução (ε) em segundos para o *SVM*.

Pode-se verificar, para a Tabela 7 que, de uma forma geral, o tempo de treinamento para o *SVM* é razoável. Ou seja, no pior caso, temos um tempo de **21.03** segundos para o experimento **10**, que obteve melhor acurácia (Tabela 6). Para os melhores casos, temos um tempo abaixo de **16** segundos (experimentos **3** e **7**). Em termos de acurácia, se voltarmos na Tabela 6, o experimento **3** esteve condizente com a média, porém o experimento **7** obteve pior resultado (tanto para o percentual de acerto quanto para Precisão (π), *F-measure* (F_β), *Estatística Kappa* (κ) e Raiz do Erro Quadrático Médio (*RMSE*)). Dessa forma, analiticamente, observa-se uma tendência em termos melhor acurácia quando o tempo de treinamento do algoritmo é maior. Quando o tempo é menor, em geral, a acurácia fica ligeiramente comprometida.

O tempo de execução do algoritmo, chega a ser irrelevante para termos de comparação, pois como o *Weka* retorna o valor de tempo com precisão de duas

(2) casas decimais, temos que para todos os experimentos o tempo t esteve no intervalo $0.01 \leq t < 0.02$.

6.2 KNN

Para a execução dos experimentos a partir do método *KNN*, utilizou-se do algoritmo *ibk* (da categoria *lazy*) que já vem, por padrão, incluso no *Weka*. Os resultados obtidos, para o referido método, encontram-se na Tabela 8.

Experimento	Acerto(%)	π	ρ	F_β	κ	<i>RMSE</i>	σ
1	70.88	0.75	0.61	0.67	0.42	0.54	0.07
2	74.01	0.76	0.71	0.73	0.48	0.51	0.06
3	71.79	0.76	0.64	0.69	0.44	0.53	0.04
4	69.46	0.72	0.65	0.68	0.39	0.55	0.04
5	68.93	0.72	0.64	0.67	0.38	0.55	0.06
6	73.75	0.79	0.65	0.71	0.48	0.51	0.05
7	68.57	0.70	0.65	0.67	0.37	0.56	0.04
8	73.93	0.77	0.69	0.72	0.48	0.51	0.05
9	72.32	0.76	0.66	0.70	0.45	0.52	0.06
10	69.46	0.73	0.62	0.67	0.39	0.55	0.06
Média	71.31	0.75	0.65	0.69	0.43	0.53	0.05

Tabela 8: Resultados da execução do *KNN*.

Em termos de percentual de acerto e *Recall* (ρ), temos que o experimento **2** obteve melhor resultado. Na sequência, temos os experimentos **8** e **6**, nessa ordem. Estes últimos estão destacados, pois a despeito do percentual de acerto, também obtiveram menor raiz do erro quadrático médio (*RMSE*), maior precisão (π) e menor desvio padrão (σ) em relação ao experimento **2**.

Na Tabela 9, podemos comparar o tempo de treinamento do modelo e tempo de execução do mesmo sobre os dados de teste não classificados para o *KNN*.

Experimento	τ (s)	ε (s)
1	20.08	0.02
2	21.64	0.01
3	23.01	0.01
4	21.98	0.02
5	21.46	0.01
6	19.87	0.01
7	22.06	0.01
8	20.36	0.02
9	22.21	0.01
10	21.76	0.01
Média	21.44	0.01

Tabela 9: Tempo de treinamento (τ) e execução (ε) em segundos para o *KNN*.

Verifica-se, de uma forma geral, o tempo de treinamento do *KNN* é maior ao do *SVM*. Para o pior caso, temos um tempo de **23.01** segundos para o experimento **3**. Para o melhor caso, temos um tempo de **19.87** segundos (experimento **6**), que em termos de acurácia, se voltarmos na Tabela 8, obteve segundo melhor resultado. Ou seja, não observa-se, claramente, uma tendência em obtermos melhor acurácia quando o tempo de treinamento do algoritmo é maior, ou o contrário.

Observa-se pequena variação no tempo de execução do algoritmo *KNN*. Mesmo nos piores casos (**1**, **4** e **7**) em que o tempo foi da ordem de **0.02** segundos, podemos afirmar que a execução se deu de forma bastante rápida.

6.3 MLP

Para a execução do algoritmo *Multilayer Perceptron (MLP)*, utilizou-se a função *MultilayerPerceptron* (da categoria *functions*) que já vem, por padrão, inclusa no *Weka*. Na sequência temos a Tabela 10, que contém os resultados obtidos para os experimentos usando o referido algoritmo.

Experimento	Acerto(%)	π	ρ	F_{β}	κ	<i>RMSE</i>	σ
1	58.25	0.61	0.41	0.38	0.16	0.60	0.08
2	61.79	0.70	0.73	0.61	0.24	0.57	0.10
3	58.21	0.50	0.42	0.38	0.16	0.63	0.09
4	58.04	0.66	0.65	0.53	0.16	0.60	0.07
5	59.11	0.57	0.47	0.44	0.18	0.58	0.11
6	58.75	0.68	0.51	0.45	0.17	0.60	0.08
7	56.43	0.73	0.42	0.42	0.13	0.62	0.07
8	57.14	0.59	0.61	0.52	0.14	0.59	0.09
9	59.82	0.59	0.61	0.54	0.20	0.56	0.09
10	68.57	0.79	0.55	0.59	0.37	0.50	0.08
Média	59.61	0.64	0.54	0.49	0.19	0.59	0.09

Tabela 10: Resultados da execução do *MLP*.

Podemos observar, claramente, que o *MLP* obteve resultados ruins, se comparado às execuções já descritas para *SVM* e *KNN*. O experimento **10** obtém melhor resultado, para as métricas: Percentual de Acerto, Precisão (π), *F-measure* (F_{β}) e *Estatística Kappa* (κ). Porém, mesmo o melhor caso, ainda está aquém dos resultados obtidos pelos métodos anteriores. Isso, provavelmente, se deve ao fato de que o algoritmo tenha sido executado com seus parâmetros de entrada padrão, conforme descrito e justificado na Subseção 5.5.3.

Em seguida, tem-se a Tabela 11 que apresenta o tempo de treinamento do modelo e o tempo de execução do mesmo sobre os dados de teste não classificados.

Podemos verificar, que além de pior acurácia, o *Multilayer Perceptron* também apresentou piores tempos de construção do modelo de classificação e pior tempo de execução.

Mesmo para o experimento **4** que obteve melhor desempenho, o tempo de **193.33 segundos** esteve, pelo menos, **8** vezes maior que o pior dos tempos de

Experimento	τ (s)	ε (s)
1	195.36	0.03
2	195.97	0.02
3	197.32	0.02
4	193.33	0.02
5	196.28	0.03
6	196.39	0.03
7	196.92	0.03
8	195.79	0.02
9	195.67	0.02
10	195.84	0.02
Média	195.89	0.02

Tabela 11: Tempo de treinamento (τ) e execução (ε) em segundos para o *MLP*.

treinamentos analisados anteriormente. Que foi o tempo de **23.01 segundos** para o experimento **3** do algoritmo *KNN*.

Quanto ao tempo de execução, os experimentos realizados com o *Multilayer Perceptron*, apesar de serem considerados rápidos, estiveram acima dos resultados encontrados para *SVM* e *KNN*. É o caso dos experimentos **1, 5, 6 e 7** que executaram em **0.03 segundos**.

6.4 *ELM*

Finalmente, temos a Tabela 12 com os dados da execução para o *Extreme Learning Machine (ELM)* que servirão de base para verificar seu desempenho em termos de acurácia e tempo de treinamento (e execução). O *ELM* foi desenvolvido seguindo o algoritmo descrito na Subseção 4.4.1 conforme metodologia presente na Subseção 5.5.4.

Observa-se que, o maior percentual de acerto (**72.58 %**) foi obtido no experimento **8** e está levemente acima da média dos demais experimentos para o método

Experimento	Acerto(%)	π	ρ	F_β	κ	<i>RMSE</i>	σ
1	69.72	0.67	0.68	0.67	0.35	0.51	0.05
2	67.89	0.67	0.62	0.64	0.32	0.54	0.07
3	69.35	0.67	0.68	0.67	0.35	0.53	0.05
4	65.71	0.66	0.66	0.66	0.31	0.55	0.05
5	68.93	0.70	0.67	0.68	0.38	0.52	0.05
6	66.96	0.67	0.65	0.66	0.34	0.52	0.06
7	68.43	0.67	0.65	0.66	0.33	0.54	0.04
8	72.58	0.73	0.73	0.73	0.45	0.49	0.07
9	67.00	0.65	0.67	0.66	0.30	0.54	0.05
10	70.36	0.70	0.73	0.71	0.41	0.50	0.07
Média	68.69	0.68	0.67	0.67	0.35	0.52	0.06

Tabela 12: Resultados da execução do *ELM*.

ELM. De forma que, os métodos anteriores, *SVM* e *KNN*, obtiveram melhores resultados, inclusive em praticamente todas as outras métricas, exceto para o *Recall* (ρ) onde, constata-se um pico de **0.73** para os experimentos **8** e **10** referente ao *ELM*.

Em comparação com o *Multilayer Perceptron*, pode-se concluir que os resultados, foram superiores. De forma que, o Percentual de Acerto, Precisão (π), *Recall* (ρ), *F-measure* (F_β) e *Estatística Kappa* (κ) estiveram acima dos encontrados pelo *MLP* (Tabela 10). Inclusive a Raiz do Erro Quadrático Médio (*RMSE*) e a Variância (π) foram menores para o *ELM*, o que são bons indicadores de confiança para tal afirmação.

A Tabela 13, traz o tempo de treinamento do modelo e tempo de execução do mesmo sobre os dados de teste não classificados.

O que podemos concluir da Tabela 13, é que, tanto o tempo de treinamento quanto o tempo de execução são evidentemente menores do que todos os métodos analisados anteriormente (*SVM*, *KNN* e *Multilayer Perceptron*). Estando, pois,

Experimento	τ (s)	ε (s)
1	0.87	0.00
2	0.68	0.00
3	0.81	0.00
4	0.64	0.00
5	0.76	0.00
6	0.72	0.00
7	0.82	0.00
8	0.82	0.00
9	0.77	0.00
10	0.76	0.00
Média	0.77	0.00

Tabela 13: Tempo de treinamento (τ) e execução (ε) em segundos para o *ELM*.

todos os tempos de treinamento abaixo de **0.88 segundos** e de execuções abaixo de **um centésimo de segundos** (como o *Weka* retorna os valores de tempo com apenas duas casas decimais, qualquer tempo abaixo dos centésimos de segundo estará como **0.00**).

Para o melhor experimento temos um tempo de treinamento de **0.64 segundos**, que é extremamente mais rápido do que os tempos de treinamento obtidos em todos os outros métodos. Mesmo para o pior experimento, no qual o *ELM*, obteve tempo de treinamento de **0.87 segundos**, este tempo ainda é, **pelo menos, 18 vezes** mais rápido que o *SVM*; **21 vezes** mais rápido que o *KNN* e **222 vezes** mais rápido que o *MLP*⁷.

Quanto ao tempo de execução do *ELM*, pode-se constatar que é muito rápido. Sendo, inclusive, mais rápido que do que todos os outros experimentos e estando abaixo dos centésimos de segundo para todos os experimentos realizados.

⁷Utilizou-se os melhores resultados do *SVM*, *KNN* e *MLP* comparados com o pior resultado para o *ELM* para o cálculo.

Após essas comparações gerais acerca dos resultados obtidos nos experimentos, apresenta-se na sequência, uma Seção destinada a discussões gerais sobre os experimentos e métodos.

6.5 Discussões gerais

Na intenção de realizar um comparativo entre os métodos, levando em consideração apenas o melhor experimento de cada método em termos de acurácia, é apresentada a seguir, a Tabela 14. O método *ELM* está em destaque por se tratar de um método novo para este tipo de aplicação e desenvolvido no presente trabalho para ser comparado com os métodos clássicos da literatura.

Método	Acerto(%)	π	ρ	F_β	κ	<i>RMSE</i>	σ	τ (s)	ε (s)
<i>SVM</i>	74.11	0.76	0.71	0.73	0.48	0.51	0.06	21.03	0.01
<i>KNN</i>	74.01	0.76	0.71	0.73	0.48	0.51	0.06	21.64	0.01
<i>MLP</i>	68.57	0.79	0.55	0.59	0.37	0.50	0.08	195.84	0.02
<i>ELM</i>	72.58	0.73	0.73	0.73	0.45	0.49	0.07	0.82	0.00

Tabela 14: Comparativo de execução para o melhor acurácia de cada método.

Primeiramente temos, o resultado do experimento **10** da Tabela 6 que foi (segundo a Tabela 7) o pior caso para o *SVM* em relação ao tempo. Pois, conforme dito anteriormente (Seção 6.1), o *SVM* apresentou uma relação inversa entre acurácia e tempo de treinamento. Portanto, para o melhor caso em termos de acurácia temos o pior tempo de treinamento (τ).

Para o *KNN* o resultado com melhor acurácia (experimento **2** da Tabela 8) obteve um tempo de treinamento mediano (**21.64** segundos), visto que os tempos de treinamento estiveram entre **19.87** segundos e **23.01** segundos. De uma forma geral, a acurácia do método *KNN* esteve bem próxima das obtidas pelo *SVM*. Já o tempo de execução (ε) e, principalmente de treinamento (τ) do *KNN* foram, no

geral, levemente maiores do que os obtidos pelo *SVM*, mas como para o experimento de melhor acurácia o *SVM* obteve pior tempo de treinamento, temos que a diferença de tempo para a melhor acurácia entre o *SVM* e o *KNN* é de, apenas, **0.61** segundos, conforme dados da Tabela 14.

O *MLP* foi o método que obteve os piores resultados em termos de acurácia. O mesmo ocorreu com os tempos de execução (ϵ) e de treinamento (τ) que, segundo a Tabela 10 estiveram consideravelmente acima de todos os experimentos realizados utilizando outros métodos.

Apesar de não ter obtido a melhor acurácia em seus resultados, o *ELM* obteve resultados bem próximos dos obtidos pelos métodos *SVM* e *KNN*, inclusive, com resultados melhores do que os obtidos pelo método *MLP*.

Para o caso de melhor acurácia do *ELM* (experimento **8** da Tabela 12), temos que os tempos de execução (ϵ) e de treinamento (τ) são extremamente menores do que os obtidos pelos outros métodos. Sendo, pelo menos, **25** vezes mais rápido que o método *SVM*, **26** vezes mais rápido que o método *KNN* e **238** vezes mais rápido que o método *MLP* quanto ao tempo de treinamento (τ) para o caso de melhor acurácia de cada método. É importante notar, que, o experimento **8** (apresentado na Tabela 14 por obter melhor acurácia) obteve o segundo pior tempo de treinamento para o *ELM*, sendo pois, de **0.82** segundos. Já o melhor experimento em termos de tempo de treinamento para o *ELM* foi o experimento **4** que ocorreu em **0.64** segundos.

Para finalizar as discussões gerais, é apresentada a Tabela 15 onde são apresentados os dados da média de execução de cada método. Pode-se perceber que os métodos *SVM* e *KNN* obtiveram médias semelhantes, visto que seus resultados foram bem parecidos. Somente para o tempo médio de geração do modelo

é que o *SVM* obteve resultado consideravelmente melhor do que o método *KNN*. Quanto ao *MLP*, sua média esteve bem aquém dos outros métodos, tanto em acurácia quanto para os tempos de treinamento e execução. Já o *ELM* demonstrou a terceira melhor média em termos de acurácia e seu tempo de treinamento médio foi **22.8** vezes menor do que o *SVM*, **27.8** vezes menor do que o *KNN* e **254.4** vezes menor do que o *MLP*.

Método	Acerto(%)	π	ρ	F_{β}	κ	<i>RMSE</i>	σ	τ (s)	ε (s)
<i>SVM</i>	71.32	0.74	0.65	0.69	0.42	0.53	0.05	17.59	0.01
<i>KNN</i>	71.31	0.75	0.65	0.69	0.43	0.53	0.05	21.44	0.01
<i>MLP</i>	59.61	0.64	0.54	0.49	0.19	0.59	0.09	195.89	0.02
<i>ELM</i>	68.69	0.68	0.67	0.67	0.35	0.52	0.06	0.77	0.00

Tabela 15: Comparativo de execução para a média de cada método.

Após visualizar e analisar os dados apresentados, pode-se realizar algumas conclusões e cogitar-se trabalhos futuros para a linha de pesquisa e desenvolvimento aqui tratada.

7 Conclusões e trabalhos futuros

Como uma técnica de aprendizagem, *ELM* tem demonstrado grande potencial para resolver problemas de classificação e regressão. Por se tratar de uma técnica recente, tem atraído considerável atenção por parte das comunidades de Inteligência Artificial e de Máquinas de Aprendizado.

Seus resultados positivos, tanto para generalização quanto para tempo de treinamento, são um diferencial em relação aos algoritmos mais utilizados (*SVM*, *KNN* e *MLP*) que se reflete em um grande motivacional para o estudo aqui apresentado. De forma, que, como demonstrado no Capítulo 6, além de boa acurácia, o método *ELM* apresenta um tempo de treinamento e de execução expressivamente reduzidos em relação aos métodos comumente utilizados.

Conclui-se que tal potencial pode ser muito bem empregado na Mineração de Texto que, segundo Hotho *et al.* (2005), é uma área interdisciplinar que envolve as áreas de Recuperação de Informação, Aprendizagem de Máquina, Estatística, Linguística Computacional e Mineração de Dados. Este cenário é propício, ainda, à utilização das técnicas e métodos de Processamento de Língua Natural o que torna natural o surgimento da Mineração de Opinião.

Outro aspecto positivo do *ELM* é que o único parâmetro que necessita ser configurado manualmente é a quantidade de neurônios na camada interna. Dessa forma a intervenção humana passa a ser baixíssima se comparada aos outros métodos, o *MLP*, onde necessita-se de ajustar 15 parâmetros.

Algumas desvantagens do *ELM*, devem ser consideradas:

1. Por se tratar de um assunto ainda recente, iniciado em 2004 por Huang *et al.* (2004), existe carência de referencial e muitos trabalhos em aberto.
2. Difícil modelagem matemática.
3. Implementação não trivial.
4. Falta de provas teórico-matemáticas para questões empiricamente evidenciadas.

Mesmo ponderando os aspectos negativos ainda temos os incentivos e justificativas para dar prosseguimento ao estudo e desenvolvimento de Máquinas de Aprendizado Extrema aplicadas à Análise de Sentimentos e Mineração de Opiniões, que constitui o objetivo principal do presente trabalho.

Como trabalhos futuros, podemos citar algumas tarefas:

- Melhoria da acurácia do algoritmo *ELM*, utilizando a técnica de *Pruning* (HUANG *et al.*, 2011).
- Implementar uma variante do *ELM* para realizar a construção do modelo de treinamento em tempo real. Esta técnica é conhecida como *online-ELM*.
- Inserção de um dicionário de palavras para ajudar o modelo durante a classificação das mensagens.
- Criação de uma aplicação comercial para realização de Análise de Sentimentos, podendo, inclusive empregar *online-ELM* caso este método obtenha bons resultados.

Referências

ARANHA, C. N.; VELLASCO, M. M. B. R.; PASSOS, E. P. L. Uma abordagem de pré-processamento automático para mineração de textos em português: Sob o enfoque da inteligência computacional. Pontifícia Universidade Católica do Rio de Janeiro-PUC- RIO, 2007.

BARTLETT, P. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *EEE Trans Inf Theory*, v. 1, p. 525–536, 1998.

CARRILHO, J. R. Desenvolvimento de uma metodologia para mineração de textos. Pontifícia Universidade Católica do Rio de Janeiro-PUC-RIO, 2007.

COFFMAN, K.; ODLYZKO, A. The size and growth rate of the internet. *First Monday*, v. 3, n. 10, out. 1998.

EGUCHI, K.; LAVRENKO, V. Sentiment retrieval using generative models. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 2006 conference on empirical methods in natural language processing*. [S.l.], 2006. p. 345–354.

EL-MANZALAWY, Y.; HONAVAR, V. WLSVM: Integrating libsvm into weka environment. 2005. Software available at <http://www.cs.iastate.edu/~yasser/wlsvm>.

ESMIN, A. A. A.; COELHO, T. A.; ABREU, A. A. A. M. de. Tcm: Twitter collector module. 2011. Laboratório de Inteligência Computacional e Sistemas Avançados (LICESA): <http://licesa.dcc.ufla.br>.

- GOLEMAN, D. Emotional intelligence: Why it can matter more than iq. v. 1, 1996.
- GUPTA, V.; LEHAL, G. A survey of text mining techniques and applications. *Journal of Emerging Technologies in Web Intelligence*, v. 1, n. 1, p. 60–76, 2009.
- HALL, M. *et al.* The weka data mining software: an update. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145. Disponível em: <<http://doi.acm.org/10.1145/1656274.1656278>>.
- HOTHO, A.; NURNBERGER, A.; PAAS, G. A brief survey of text mining. *Machine Learning*, v. 20, n. 1, p. 19–62, 2005.
- HUANG, G.; L., C.; SIEW, C. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *EEE Trans Neural Network*, v. 1, p. 879–892, 2006.
- HUANG, G.; SIEW, C.; ZHU, Q. Extreme learning machine. *Technical Report ICIS/03/2004*, jan. 2004.
- HUANG, G.; WANG, D.; LAN, Y. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, Springer, p. 1–16, 2011.
- HUANG, G. B.; ZHU, Q.; SIEW, C. Extreme learning machine: a new learning scheme of feedforward neural networks. *Proceedings of international joint conference on neural networks (IJCNN2004), Budapest, Hungary*, v. 2, p. 985–990, 2004.
- LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. *Biometrics*, International Biometric Society, v. 33, n. 1, p. 159–174, 1977. Disponível em: <<http://www.ncbi.nlm.nih.gov/pubmed/843571>>.

LIN, F. ren; HSIEH, L. shih; CHUANG, F.-T. Discovering genres of online discussion threads via text mining. *Computers And Education*, v. 52, n. 2, p. 481–495, 2009.

LOWE, D. Adaptive radial basis function nonlinearities and the problem of generalisation. *Proceedings of first IEE international conference on artificial neural networks*, v. 1, p. 171–175, 1989.

MANNING, C. D.; RAGHAVAN, P.; SCHATZ, H. Introduction to information retrieval. Cambridge University Press, New York, NY, USA, 2008. Disponível em: <<http://portal.acm.org/citation.cfm?id=1394399>>.

MITCHELL, T. The discipline of machine learning. *Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University*, 2006.

OPSYS. Mineração de opinião em conteúdo web - acessado em agosto de 2012. 2012. URL: <http://www.opsys.com.br/>.

PANG, B.; LEE, L. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, v. 2, n. 1-2, p. 1–135, 2008.

PRIA, A. D. The development of formal grammars under lexicalist hypothesis and its association to requirements of nlp systems. *Campus de Alto Araguaia, Depto. de Letras, Universidade do Estado de Mato Grosso*, v. 1, 2008.

RAUBER, T. W. Redes neurais artificiais. 2005.

RUMELHART, D.; HINTON, G.; WILLIAMS, R. Learning representations by back-propagation errors. *Nature*, v. 2, p. 533–536, 1986.

RUMELHART, D.; RAMSEY, W.; STICH, S. Philosophy and connectionist theory. *Lawrence Erlbaum*, v. 1, 1991.

SALTON, W. e. Y. A vector space model for automatic indexing. *Commun. ACM, ACM, New York, NY, USA*, v. 3, n. 18, p. 613–620, 1975.

SENTWEET. Acessado em agosto de 2012. 2012. URL: <https://github.com/azuranop/sentweet>.

SIEW, C.; HUANG, G.; ZHU, Q. Extreme learning machine: Rbf network case. *Proc. Int. Conf. Control Autom. Robot. Vis*, p. 1651–1663, dez. 2004. <Http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm>.

TEXTMININGNEWS. Acessado em junho de 2011. 2011. URL: <http://www.textminingnews.com>.

VELLASCO, M. M. B. R. Redes neurais artificiais. 1994.

VIGOTSKI, L. A construçao do pensamento e da linguagem. Martins Fontes, 2001.

VIGOTSKI, L.; LURIA, A. R. Introduction to the russian translation of freud's beyond the pleasure principle. *The Vygotsky reader. Oxford & Cambridge*, 1994.

WOLFRAMALPHA. Acessado em agosto de 2012. 2012. URL: <http://www.wolframalpha.com/>.

ZAGHLOUL, W.; LEE, S.; TRIMI, S. Text classification: neural networks vs support vector machines. *Industrial Management & Data Systems*, Emerald Group Publishing Limited, v. 109, n. 5, p. 708–717, 2009.