PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIAS

FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

PEDRO PAULO MONTEIRO

FILTRAGEM DE PÁGINAS WEB BASEADA EM REDES NEURAIS ARTIFICIAIS DE HOPFIELD

CAMPINAS

2007

PEDRO PAULO MONTEIRO

FILTRAGEM DE PÁGINAS WEB BASEADA EM REDES NEURAIS ARTIFICIAIS DE HOPFIELD

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Pontifícia Universidade Católica de Campinas, como exigência da disciplina Projeto Final II.

Orientador: Prof. Dr. Carlos Miguel Tobar Toledo.
Co-Orientador: Prof. Dr. Juan Manuel Adán Coello.

CAMPINAS

2007

BANCA EXAMINADORA

Presidente e Orientador Prof. Dr. Carlos Miguel Tobar Toledo

1º Examinador Prof. Dr. Juan Manuel Adán Coello

Campinas, 04 de Dezembro de 2007

Dedico este trabalho aos meus amados pais, José Américo Monteiro e Maria de Fátima C. S. Monteiro que foram alicerces na construção da minha formação humana.

AGRADECIMENTOS

Ao grande Deus-Pai,
Minha fonte de vida e equilíbrio espiritual, por prover-me capacidade intelectual para a realização deste trabalho.

Ao Prof. Dr. Juan Manuel Adán Coello,
Co-Orientador deste trabalho que prestou assistência fundamental para o seu desenvolvimento.

Ao Prof. Dr. Carlos Miguel Tobar Toledo,
Orientador que dirigiu o andamento metodológico do trabalho com bastante seriedade.

À Srta. Priscila Vieira Franco,
Namorada que me apoiou incondicionalmente, por sua compreensão, companheirismo e afeto.

RESUMO

MONTEIRO, Pedro Paulo. *Filtragem de Páginas Web baseada em Redes Neurais Artificiais de Hopfield.* Campinas, 2007. 51f. Trabalho de Conclusão de Curso (Graduação) - Curso de Engenharia de Computação, Pontifícia Universidade Católica de Campinas. Campinas, 2007.

No contexto em que a *World Wide Web* e o seu repositório de informação cresce de maneira ininterrupta, as ferramentas de busca amplamente utilizadas pelos usuários da Internet desempenham um papel essencial para a tarefa de recuperação de informação relevante. Entretanto, há neste contexto o problema da não garantia de relevância dos resultados apresentados por essas ferramentas. Esta é a motivação para o desenvolvimento de aplicações capazes de filtrar de forma eficaz as informações da rede. Logo, este trabalho tem o objetivo de aumentar a relevância dos resultados das buscas. Para tanto, este trabalho aborda uma proposta baseada em Redes Neurais Artificiais de Hopfield, em conjunto com técnicas de indexação automática e geração de espaço de conceitos para a filtragem de páginas da *Web* de forma mais aderente aos propósitos dos usuários. A aplicação resultante deste trabalho foi concebida sob a metodologia de projeto *Scrum,* sendo que a linguagem de programação usada foi a Java. Para a avaliação do trabalho, foram utilizadas as métricas de Precisão e Cobertura, a serem medidas em experimentos de filtragens de uma coleção de documentos que tratam do assunto música brasileira.

Termos de indexação: redes neurais artificiais, filtragem de páginas web, indexação automática, geração de espaço de conceitos.

ABSTRACT

MONTEIRO, Pedro Paulo. *Web Pages Filtering based on Hopfield Artificial Neural Networks.* Campinas, 2007. 51f. Trabalho de Conclusão de Curso (Graduação) - Curso de Engenharia de Computação, Pontifícia Universidade Católica de Campinas. Campinas, 2007.

In a context where the World Wide Web and its information repository grows in a unstoppable way, the widely used web search tools play an essential role for their users, in order to perform the task of retrieving relevant information from the Internet. However, there is a problem in that context, which is the lack of relevance on the results presented by those tools. That is the motivation for developing applications capable of filtering Web information effectively. Therefore, this work has the objective of raising the relevance of the search results. For that, this work presents a system based on the Hopfield Artificial Neural Network combined with automatic indexing and concept space generation methods for filtering Web pages according to users' purposes. The resulting application of this work was conceived under the Scrum projects methodology and the Java programming language was used. For assessment of this work, the Precision and Recall metrics were planned in filtering experiments over a document collection the about brazilian music subject.

Index terms: artificial neural networks, web pages filtering, automatic indexing, concept space generation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	=	Application Programming Interface
CPL	=	Common Public License
HTML	=	Hyper Text Markup Language
MLP	=	Multilayer Perceptron
NILC	=	Núcleo Interinstitucional de Lingüística Computacional
WWW	=	World Wide Web
RNA	=	Rede Neural Artificial
TXT	=	Text File
RI	=	Recuperação da Informação
UML	=	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Desde sua criação, a *World Wide We*b (WWW) apresenta taxas de crescimento espantosas. Isso se deve ao fato da sua alta acessibilidade e escabilidade que propiciam um ambiente muito favorável para o compartilhamento de informações entre usuários. Segundo um levantamento realizado em setembro de 2007 por um provedor de serviços norte-americano chamado *Netcraft*[1], estima-se que existam mais de 135,1 milhões de sítios na *Web*. Há também uma projeção que indica que a taxa de crescimento atual da *Internet* levará à existência de aproximadamente duzentos milhões de sítios no ano de 2010.

Em virtude desta realidade, surge a problemática da recuperação dos documentos na rede pelo usuário. Neste cenário, as ferramentas de busca na *Web* como o *Google*[2], por exemplo, desempenham um importante papel dada a enorme quantidade de documentos disponíveis, pois elas apresentam uma maneira fácil e eficiente para a execução de pesquisas. Não obstante, ainda assim, é difícil a busca eficaz de conteúdos relevantes, pois em muitas situações o usuário precisa realizar uma pós-filtragem onerosa dos resultados obtidos em suas consultas.

É justamente neste contexto que este trabalho se insere. Seu objetivo é melhorar a eficácia dos resultados das ferramentas de busca, baseando-se em Redes Neurais Artificiais de Hopfield. O modelo proposto provê ao usuário uma forma de definir conceitos de seu interesse e assim contextualizar suas buscas, de forma que a filtragem dos documentos se dá pelo reconhecimento de padrões armazenados daqueles conceitos pela Rede Neural Artificial.

A seguir é detalhado e caracterizado o problema abordado, o objetivo do trabalho, o estado da arte e a estrutura desta monografia.

[1] http://news.netcraft.com
[2] http://www.google.com

1.1 Caracterização do Problema

Há um problema intrínseco ao estado atual da maioria das ferramentas de busca, pois apesar de proverem uma grande quantidade de resultados de forma eficiente, não garantem a relevância das informações. Isso ocorre, em alguns casos, pela contextualização limitada desempenhada pelas ferramentas de buscas, que utilizam para a filtragem dos documentos apenas a ocorrência do conjunto de palavras-chave informadas pelo usuário.

Logo, este trabalho busca a melhoria na qualidade da filtragem nas buscas por documentos na *Web*, uma vez que, se for considerado o inúmero acervo de páginas na *Internet*, que cresce ininterruptamente, torna-se cada vez mais difícil realizar buscas contextualizadas e selecionar documentos pertinentes a um assunto de interesse, ou seja, de forma eficaz.

1.2 Objetivo

Devido a este trabalho ser relacionado à área da Recuperação de Informação (RI), o objetivo foi em termos de desenvolver um sistema que aumente a relevância dos resultados das filtragens de documentos, sendo o mesmo capaz de identificar os interesses dos usuários. Para avaliar o sistema e o alcance deste objetivo, são usadas as métricas de Precisão e de Cobertura.

Precisão e Cobertura são taxas que constituem um dos critérios amplamente utilizados na área da Ciência da Informação para avaliar quantitativamente a eficácia dos sistemas de Recuperação de Informação.

A taxa de Precisão consiste na relação entre a quantidade de informações corretas obtidas e o número total de informações (corretas e incorretas). Já taxa de Cobertura determina a relação entre o total de informações corretas obtidas e o total de informações corretas presentes nos documentos considerados.

1.3 Estado da Arte

O trabalho de MARIN (2003) aborda um sistema para a filtragem de páginas da *Web*, baseado no modelo de Hopfield, assim como é feito neste Trabalho de Conclusão de Curso. Aquele trabalho também utiliza as técnicas de indexação automática de documentos, a geração de espaço de conceitos e a RNA de Hopfield para a filtragem.

O ponto de diferença nas técnicas mencionadas, é que na etapa de indexação automática de documentos, detalhada na seção 2.1, foi adicionado o processo de lematização dos termos, com a intenção de melhoria dos resultados das filtragens, neste Trabalho de Conclusão de Curso.

Além disso, outra diferença é que o trabalho anterior não foi concebido sob uma arquitetura adequada para a *Web*, que permita a filtragem de páginas presentes em repositório *on line*. Diferentemente, neste Trabalho de Conclusão de Curso, é possível a filtragem de páginas *on line*, graças ao uso da Yahoo Search API, apresentada na seção 4.1.1.

Um segundo trabalho semelhante é o de NETO e VIEIRA (2005), que apresenta uma implementação de um mecanismo de Recuperação de Informação na *Web,* baseado em redes neurais no modelo *MLP* (*multilayer perceptron*). Sua proposta é uma análise da similaridade dos textos com a retroalimentação ou Retorno de Relevância da RNA.

A solução difere do trabalho atual, no modo como o usuário interage para a determinação da relevância dos documentos. Naquele trabalho, após uma busca inicial, o usuário indica quais dos documentos resultantes são relevantes, a fim de que seja efetuado um treinamento contínuo da RNA para as buscas futuras.

Já no atual trabalho, antes de efetuar buscas, o usuário deve selecionar documentos relevantes de exemplo, para o aprendizado da RNA sobre a coleção escolhida. O aprendizado é estático e a única forma para se refinar o treinamento da RNA é reapresentando documentos mais relevantes ao sistema.

1.4 Estrutura da Monografia

No decorrer desta monografia, são apresentadas as atividades desenvolvidas durante o Trabalho de Conclusão de Curso. A mesma é divida em Capítulos. A seguir está uma visão geral sobre a abordagem de cada um deles:

- Referencial Teórico: traz subsídios e faz referências sobre os algoritmos e conceitos utilizados no desenvolvimento;

- Metodologia: traz referência e aborda os detalhes sobre os métodos de trabalho utilizados;

- Aplicação para Filtragem: descreve o histórico de todo o desenvolvimento do trabalho, referente às etapas realizadas para a implementação do software;

- Avaliação e Validação: apresenta o planejamento dos experimentos a serem realizados para a verificação do alcance do objetivo definido;

- Conclusão: relata as conclusões das etapas do trabalho, bem como limitações e encaminhamentos para trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Indexação Automática de Documentos

Na área de Recuperação da Informação (RI), entende-se por indexação de documentos o processo de seleção de palavras ou termos que melhor representem seus conteúdos e características. Ao final do processo, espera-se que as palavras-chave extraídas possam facilitar a recuperação da informação de tais documentos. Em outras palavras, esse processo significa criar um resumo para o documento em questão.

No entanto, a Ciência da Computação estuda técnicas de automatizar esse processo, elaborando e combinando algoritmos para que seja possível uma indexação automática dos documentos. Os passos para a realização da indexação automática são os seguintes:

- Identificação das palavras do documento: são extraídos do documento apenas os conjuntos de caracteres alfabéticos consecutivos, que possam representar palavras da língua portuguesa. Para isso, se faz necessária a remoção de todas as *tags* HTML (Hyper Text Markup Language) das páginas e os caracteres não alfabéticos, como pontos, vírgulas, hífens, parênteses, colchetes e vários outros possíveis símbolos. Neste processo também é realizada a padronização do caixa do texto (transformação de todas as letras em maiúsculas ou minúsculas).

Para exemplificar o funcionamento do processo de identificação das palavras, a Figura 1 mostra o conteúdo de um documento HTML de entrada antes dos procedimentos mencionados e a Figura 2 mostra a representação das palavras resultantes identificadas.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Uma Introdução às Redes Neurais </title></head>

<body link="#00FFFF" vlink="#800080">

<p align="center"><img src="Head.jpg" width="148" height="162"></p>

<h1 align="center">
    <font color="#0000FF">
        Seja bem vindo à página de Redes Neurais!
    </font></h1>

<p align="center">
<font size="4">
    Aqui você terá noções básicas de redes neurais, passando por seu
    histórico, topologias, suas aplicações, passos para se
    desenvolver aplicações utilizando conceitos de redes neurais,
    chegando até exemplos práticos desenvolvidos por empresas
    espalhadas pelo mundo todo e que podem ser visitadas pela
    internet!
</font></p>

</body>
</html>
```

Figura 1. Documento HTML antes do processo de identificação de palavras.

```
uma introdução às redes neurais seja bem vindo
à página de redes neurais aqui você terá noções
básicas de redes neurais passando por seu histórico
topologias suas aplicações passos para se desenvolver
aplicações utilizando conceitos de redes neurais
chegando até exemplos práticos desenvolvidos por
empresas espalhadas pelo mundo todo e que podem ser
visitadas pela internet
```

Figura 2. Representação das palavras identificadas no documento HTML.

- Remoção de *stopwords*: é a remoção de palavras que não agregam relevância ao conteúdo do documento, pois estão presentes em qualquer documento. Essas palavras são denominadas *stopwords* e incluem, por exemplo, artigos, preposições, adjetivos e pronomes. A Figura 3 apresenta a indicação das *stopwords* a serem removidas dentre as palavras identificadas no exemplo anterior da Figura 2.

~~uma~~ introdução ~~às~~ redes neurais seja ~~bem~~ vindo
~~à~~ página ~~de~~ redes neurais ~~aqui~~ ~~você~~ terá noções
básicas ~~de~~ redes neurais passando ~~por~~ ~~seu~~ histórico
topologias ~~suas~~ aplicações passos ~~para~~ ~~se~~ desenvolver
aplicações utilizando conceitos ~~de~~ redes neurais
chegando ~~até~~ ~~exemplos~~ práticos desenvolvidos ~~por~~
empresas espalhadas ~~pelo~~ mundo todo ~~e~~ ~~que~~ podem ~~ser~~
visitadas ~~pela~~ internet

Figura 3. Indicação das *stopwords* a serem removidas.

- Lematização das palavras: processo para a redução das palavras aos seus radicais, pois se sabe que os mesmos representam melhor o significado das palavras, uma vez que a língua natural apresenta flexões, sufixos e inúmeras variações nas palavras. A Figura 4 mostra o resultado da lematização das palavras da Figura 3.

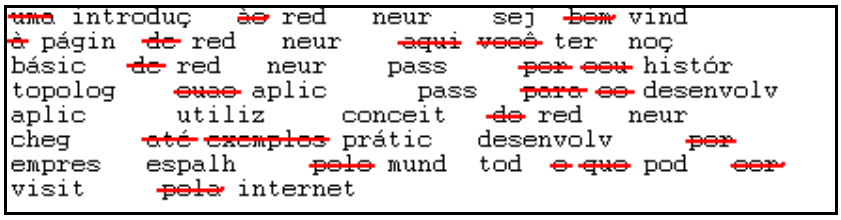

Figura 4. Resultado da lematização das palavras.

- Formação de termos: termos são composições de duas ou mais palavras adjacentes encontradas nos documentos. Isso pode ser feito, na medida em que estudos feitos na Ciência da Computação comprovam que termos com mais de uma palavra têm maior significado para representação do conteúdo de um documento. Os termos são compostos por duas a três palavras adjacentes encontradas no documento, incluindo também a formação de termos de única palavra. Exemplos de termos formados a partir das palavras lematizadas da Figura 4 são:

- Termos compostos por uma palavra: “introduç”, “red”, “neur” e “págin”.

- Termos compostos por duas palavras: “introduç-red”, “red-neur”, “vind-págin” e “histór-topolog”.

- Termos compostos por três palavras: “introduç-red-neur”, “noç-básic-red”, “histór-topolog-aplic”, “conceit-red-neur” e “pratic-desenvolv-empres”.

Ao final do processo de indexação automática, também devem ser calculadas as freqüências dos termos (*tf*), ou seja, a contagem do número vezes que os termos aparecem em cada documento da coleção. É necessário também o cálculo da freqüência de documento (*df*) para cada termo, que é a contagem de

documentos nos quais o termo ocorre. Estas freqüências são utilizadas na etapa de geração de espaço de conceitos, apresentada na seção 2.2.

2.2 Geração de Espaço de Conceitos

Apesar do processo de indexação automática, descrito anteriormente, resultar na seleção dos termos que melhor representam os documentos, as palavras soltas extraídas dos textos perdem o contexto expressado por seus autores. Para solucionar este problema, existem diversas propostas no ramo da Recuperação da Informação que apresentam técnicas para a criação de um espaço de conceitos. Um espaço de conceitos é uma representação lógica do conteúdo relevante e do contexto de uma coleção de documentos.

A geração de um espaço de conceitos tem o intuito de utilizar os termos selecionados dos documentos na etapa de indexação automática, para a geração de uma matriz que represente probabilisticamente a relação de co-existência entre os pares de termos. Estes valores probabilísticos são denominados de coeficientes assimétricos de similaridade e os mesmos são usados para representar o espaço de conceitos de uma coleção de documentos.

Porém, antes do cálculo desses valores probabilísticos, nesta etapa é realizada uma seleção dos termos gerados no processo de indexação automática, utilizando-se os fatores *df* e *tf* calculados naquele processo. A seleção pode ser feita através de dois critérios:

- Pela faixa da freqüência (*df*): são escolhidos os termos cujas freqüências (*df*) se situam entre $\frac{N}{100}$ e $\frac{N}{10}$, onde N é a quantidade de documentos da coleção.

- Pela importância do termo (d_j): A expressão matemática mostrada na equação 2.1 abaixo calcula a importância do termo na coleção de documentos.

$$d_j = tf_j \times \log\left(\frac{n}{df_j} \times NP\right) \quad (2.1)$$

Nesta expressão, *tf* é a freqüência do termo *j* na coleção de documentos; df_j é a freqüência de documento do termo *j* na coleção, ou seja, a quantidade de documentos no qual o termo *j* ocorre; *NP* representa a quantidade de palavras que compõe o termo, e *n* é a quantidade de documentos da coleção.

O fator *tf* da Equação 2.1, que é a freqüência do termo *j* na coleção de documentos, é calculado através da Equação 2.2 abaixo:

$$tf_j = \sum_{i=1}^{n} tf_{ij} \quad (2.2)$$

Onde tf_{ij} é a freqüência do termo *j* no documento *i.*

Após o cálculo da importância de cada termo, pode-se então classificá-los e selecionarem-se os termos que melhor representam o espaço de conceitos.

Porém, em ambos os critérios de seleção mencionados, pela inconveniência da seleção de um número excessivo de termos, pois uma coleção de documentos pode conter milhares de termos, deve-se também ser parametrizado o número máximo de termos a serem selecionados.

Logo, para o critério feito pela freqüência do termo, são selecionados aleatoriamente os n_{max} (número máximo de termos) termos dentro da faixa de freqüência calculada. Já no segundo critério, que é baseado na importância do termo, são selecionados os n_{max} com as maiores importâncias calculadas.

Após o processo de seleção dos termos, seguindo um dos critérios mencionados anteriormente, para o início dos cálculos dos coeficientes assimétricos de similaridade, é necessário o cálculo dos fatores d_{ij} e $d_{ijk.}$

O fator d_{ij} representa o peso do termo *j,* no documento *i.* Ele é calculado conforme a Equação 2.3, de forma semelhante à Equação 2.1.

$$d_{\bar{v}} = tf_{\bar{v}} \times \log \frac{n}{df_j} \quad (2.3)$$

Onde tf_{ij} é a freqüência do termo *j* no documento *i,* df_j é a freqüência de documento do termo *j* na coleção e *n* é a quantidade de documentos da coleção.

O fator d_{ijk} é o peso do par de termos *j e* k no documento *i.* O mesmo é calculado conforme a Equação 2.4 abaixo:

$$d_{ijk} = tf_{ijk} \times \log \frac{n}{df_{jk}} \qquad (2.4)$$

Onde tf_{ijk} é o número de ocorrência simultânea dos termos *j e* k no documento *i,* e df_{jk} é o número de documentos nos quais os termos *j* e *k* ocorrem simultaneamente, na coleção de *n* documentos.

Após o cálculo dos fatores d_{ij} e d_{ijk}, é então possível o cálculo dos coeficientes assimétricos de similaridade. As Equações 2.5 e 2.6 são a função de agrupamento assimétrica criada por CHEN (et al, 1994).

$$T_{kj} = \frac{\sum_{i=1}^{n} d_{ijk}}{\sum_{i=1}^{n} d_{ik}} \qquad (2.5) \qquad T_{jk} = \frac{\sum_{i=1}^{n} d_{ijk}}{\sum_{i=1}^{n} d_{ij}} \qquad (2.6)$$

A Equação 2.5 refere-se ao cálculo do coeficiente assimétrico de similaridade do termo *j* em relação ao termo *k,* enquanto que a Equação 2.6 ao coeficiente assimétrico de similaridade do termo *k* em relação ao termo *j.*

Logo, ao final desta etapa, obtêm-se para cada par de termos, os coeficientes assimétricos de similaridade, ou seja, uma matriz de probabilidades de co-existências entre os termos para a representação do espaço de conceitos. Essa representação é a fonte de aprendizado da Rede Neural Artificial de Hopfield, detalhada na seção 2.3.

2.3 Redes Neurais Artificiais de Hopfield

A respeito do conceito de Rede Neural Artificial (RNA):

> "As RNAs constituem uma das várias linhas de pesquisa no campo da Inteligência Artificial (IA) e têm por objetivo investigar a possibilidade da simulação de comportamentos inteligentes através de modelos baseados na estrutura e funcionamento do cérebro humano. Estes modelos são construídos a partir de técnicas computacionais e podem ser implementadas em hardware ou software. O estudo das RNAs é um dos ramos da IA que mais se desenvolve, atraindo pesquisadores de diversas áreas do conhecimento" (MARIN, 2003).

Na literatura existem diversas propostas de modelos de RNAs. Cada um deles remete a soluções diferentes, adequadas para problemas de cenários específicos. Neste trabalho, o modelo da RNA de Hopfield foi o utilizado. Basicamente, este modelo apresenta uma rede neural artificial de camada única, onde as entradas dos neurônios são realimentadas com suas saídas, com um atraso de tempo.

A comunidade científica comumente utiliza-se do modelo de Hopfield para fins de solução de problemas relacionados à recuperação de padrões armazenados, que é o caso em que este trabalho também se insere. Por exemplo, o trabalho de CHUNG, POTTENGER e SCHATZ (1998) utiliza a RNA de Hopfield para permitir a geração de índices de tópicos para um documento de forma automática, a partir do conhecimento armazenado na rede neural.

Neste trabalho, ela é usada para o aprendizado artificial dos espaços de conceitos produzidos na etapa de geração de espaço de conceitos e também para o posterior reconhecimento dos padrões contidos nos documentos a serem filtrados, a fim de se determinar a relevância deles em relação ao espaço de conceitos considerado numa busca.

A RNA possui a característica de ser uma memória do tipo associativa, ou seja, é capaz de recuperar o conhecimento armazenado a partir de partes da informação. Isso significa que dado um padrão aprendido, ao se apresentar entradas incompletas em relação ao padrão, a característica associativa permite a inferência do restante da informação.

Antes de ser apresentada a estrutura de uma RNA de Hopfield, a Figura 5 mostra um diagrama esquemático do neurônio artificial da mesma.

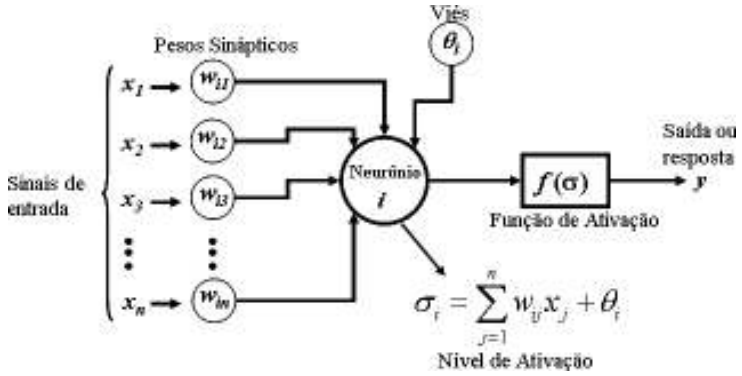

Figura 5. Diagrama esquemático de um neurônio artificial (retirado de MARIN, 2003).

Os neurônios de uma RNA de Hopfield são todos ligados uns aos outros. Logo, os sinais de entrada x_i de um neurônio, vistos na Figura 5, são os sinais de saída y, provenientes dos outros neurônios da rede, ou seja, os neurônios se realimentam.

Para que um neurônio calcule a sua saída, os sinais de entrada, juntamente com o viés (valor usado para o refinamento da ativação do neurônio), sofrem uma soma ponderada, considerando os pesos w_{in} para cada entrada.

Esta soma resulta no Nível de Ativação do neurônio. Este valor é então submetido a uma Função de Ativação. Esta função tem o intuito de determinar a saída do neurônio, indicando se o mesmo estará ativo ou não, mediante às entradas apresentadas. A função de ativação utilizada neste trabalho foi a sigmóide. A Figura 6 mostra a expressão matemática da função sigmóide usada.

$$y_j(t+1) = fs(net_j) = \frac{1}{1+\exp\left[\frac{-(net_j - \theta_1)}{\theta_0}\right]}, 1 \le j \le n$$

Figura 6. Função de ativação sigmóide de um neurônio (retirado de MARIN, 2003).

Onde $fs(net_j)$ é a função sigmóide, θ_1 e θ_0, são respectivamente, o viés de entrada no neurônio e uma constante de curvatura (inclinação da curva) da

função sigmóide, ambos são definidos empiricamente. O valor net_j é o Nível de Ativação, cujo cálculo é mostrado na Figura 5 anterior.

Após a etapa de geração de espaço de conceitos, a matriz de coeficientes assimétricos de similaridade é usada para a inicialização da RNA de Hopfield. Isso é feito, de forma que cada termo presente no espaço de conceitos representa um neurônio da RNA. Já os valores de peso das conexões entre os neurônios, ou seja, entradas e saídas, são os coeficientes assimétricos de similaridade calculados para os pares de termos. Pode-se ver na Figura 7 a representação de uma RNA de Hopfield exemplo, composta por três neurônios.

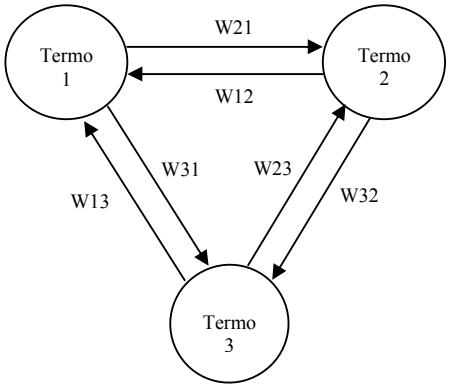

Figura 7. Exemplo de uma RNA de Hopfield composta por três neurônios.

Na Figura 7, os valores de $w_{ij,}$ são os pesos das conexões entre os neurônios, ou seja, os coeficientes assimétricos de similaridade. Daí a explicação para a denominação de assimétricos, pois os pesos das conexões de entrada podem diferir dos pesos das conexões de saída dos neurônios. Isso acontece porque na área de RI, sabe-se que a probabilidade, seguindo a Figura 7, do Termo 1 (valor w31) na presença do Termo 3, não é necessariamente a mesma no sentido inverso, representada pelo peso w13.

Para a ativação da RNA para a filtragem de documentos, no modo de filtragem, os neurônios cujos termos foram encontrados nos documentos submetidos ao processo de indexação automática são inicializados como ativos. A partir de então, interações para a convergência da RNA acontecem, de forma que os neurônios se realimentam até a estabilização da rede. Ao final da

estabilização, uma quantidade de neurônios ativos será indicada. Esta quantidade é então associada a relevância do documento em questão.

3 METODOLOGIA

A metodologia utilizada para o desenvolvimento da aplicação foi a *Scrum*[3]. Ela está fundamentada no *Agile Manifesto*[4] que prega alguns princípios no processo de desenvolvimento de software (BEEDLE et. al., 2001):

- Satisfação do cliente, através de uma rápida e continua entrega de software útil.

- A principal medida de evolução é a existência de software em funcionamento.

- Mudanças de requisitos, mesmo que tardias, são bem vindas.

- Adaptação regular à mudança de circunstâncias.

A Scrum é orientada a ciclos interativos chamados *Sprints*. Um *Sprint* nada mais é do que um período de tempo em que se mobilizam esforços para a conclusão de um conjunto de itens de *backlog* para um protótipo a ser entregue. Entende-se por *backlog,* quaisquer itens pendentes que precisam ser concluídos no projeto. Os mesmos podem abranger desde de uma atividade de levantamento de requisitos até uma atividade de codificação.. Geralmente os períodos de tempo dos *Sprints* são curtos, variam de duas semanas a um mês. Após o desenvolvimento de um *Sprint*, faz-se uma revisão dos itens de *backlog* propostos inicialmente, comparando-os com as funcionalidades que foram efetivamente resultantes.

Desta forma, planejaram-se quatro *Sprints* para o desenvolvimento da aplicação. Nos mesmos, tendeu-se a fechar versões do software funcionando e cada vez mais aderentes aos requisitos do produto final. Além disso, dentro deste trabalho, a metodologia propiciou meios de atingir mais produtividade com a diluição das atividades ao longo do cronograma. Ela estabelece cinco atividades definidas a seguir.

[3] http://www.scrumalliance.org
[4] http://en.wikipedia.org/wiki/Scrum_(development)

- Revisão dos planos da distribuição (versão do protótipo).

- Distribuição, revisão e ajuste dos padrões aos quais o produto irá aderir (SCHWABER, 2006).

- *Sprint*: A fase de desenvolvimento é um ciclo iterativo de trabalho de codificação. O gerenciamento determina em quanto tempo, qualidade ou quais funcionalidades são alcançadas. Quando as iterações são completadas a fase de fechamento ocorre (SCHWABER, 2006).

- Revisão do *Sprint*: Revisão do software desenvolvido. Comparação das mudanças implementadas com as tarefas pendentes, edição e inclusão de novos itens de *backlog*, distribuição e planejamento da próxima revisão.

- Fechamento: Quando a equipe de projeto percebe que as variáveis de tempo, requisitos, custo e qualidade concorrem por uma nova versão a ocorrer, declara-se o desenvolvimento fechado e entra-se nesta fase de fechamento. Esta fase prepara o produto desenvolvido para a distribuição geral. Integração, testes do sistema, documentação para o usuário, preparação de material de capacitação e preparação de material de propaganda então são produzidas nesta fase (SCHWABER, 2006).

A Figura 8 apresenta o ciclo iterativo *Scrum*:

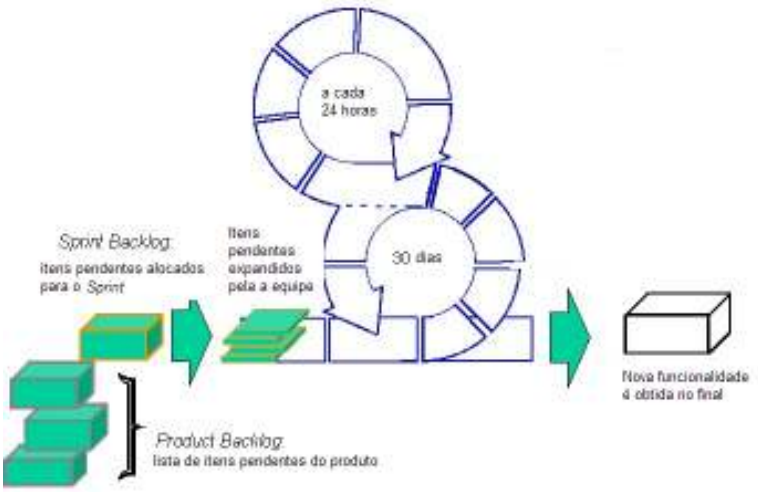

Figura 8. Processo de desenvolvimento na metodologia Scrum (retirado e adaptado de MOKABYTE, 2004).

Por se tratar de uma metodologia voltada para equipes de projeto, as revisões e reuniões, previstas na metodologia que envolveriam outros participantes, não foram feitas. Porém, todos os itens referentes ao desenvolvimento propriamente do *Sprint,* foram conduzidas individualmente pelo aluno.

Para melhor entendimento do processo da metodologia e como ela foi aplicada neste trabalho, é necessária a explicação de alguns artefatos gerados na mesma. O guia (CONCHANGO, 2006) apresenta as seguintes definições desses artefatos:

- O *Product Backlog* é uma lista priorizada de itens pendentes do projeto com os tempos estimados usados no trabalho. A idéia é que os mesmos sejam transformados em funcionalidades do produto. As estimativas são feitas em dias e quanto mais prioritário for um item, mais no topo da lista ele se encontrará. A prioridade deve ser determinada baseada nos itens de maior valor para o negócio ou que oferecem o maior retorno ao investimento no menor prazo. Esta lista deve evoluir, mudando conforme as condições de negócio ou as mudanças tecnológicas. A flexibilidade de alterações nessa lista permitiu a adequação do processo do projeto em função do andamento real do mesmo, conforme as necessidades. As principais mudanças são reladas nos *Sprints* 2 *e* 3, apresentados nas seções 4.3 e 4.4, respectivamente.

- O diagrama de *Product Burndown* é feito para dar uma indicação de quão rápido a equipe está “queimando” os itens pendentes do projeto durante o trabalho e entrega os itens do *Product Backlog*. Ele foi usado para planejar quando seriam feitas as entregas ou quando seriam removidos itens de uma entrega se o processo não fosse rápido o suficiente.

- O *Sprint Backlog* é uma lista de tarefas que define o trabalho da equipe para um *Sprint*. A lista emerge durante o planejamento do *Sprint*. As tarefas são o que a equipe definiu como sendo o necessário para transformar os itens do *Product Backlog* em uma funcionalidade do sistema. Cada tarefa identifica quem é responsável, e neste caso todas foram atribuídas ao próprio aluno.

4 APLICAÇÃO PARA FILTRAGEM

As primeiras atividades para o desenvolvimento deste trabalho envolveram o estudo da dissertação de MARIN (2003) que descreve o funcionamento do mecanismo que usa as técnicas de RI, indexação automática de documentos e geração de espaço de conceitos para filtragem de páginas *Web* baseado no modelo de Hopfield utilizado.

Com o encaminhamento do Prof. Co-Orientador, foi possível, na etapa inicial, a introdução e compreensão do tema, bem como o esclarecimento sobre a estrutura de funcionamento do software construído. Portanto, após uma investigação preliminar do tema escolhido, pôde-se modelar e planejar uma arquitetura para o software de forma que o mesmo foi divido em módulos. A Figura 9 apresenta o diagrama de arquitetura do software, resultado do estudo feito, com a indicação de tais módulos e entidades relacionadas.

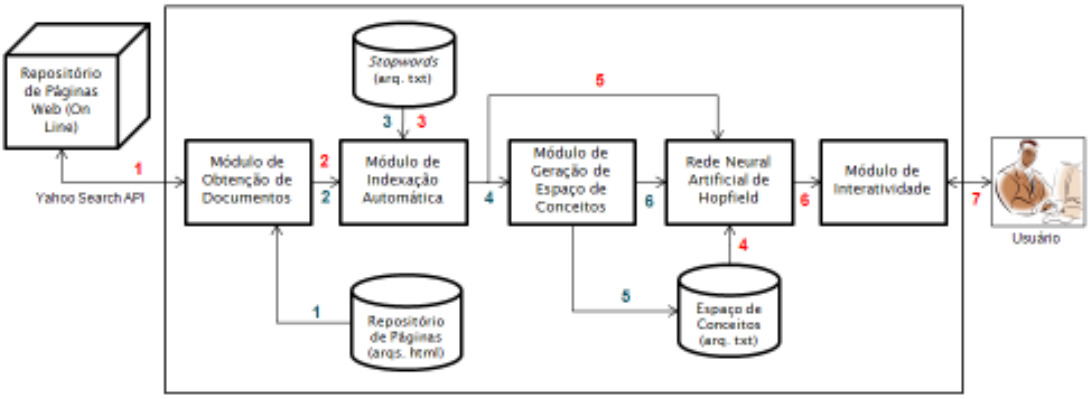

Figura 9. Diagrama de Arquitetura da aplicação.

A solução definida refletida por essa arquitetura apresenta a característica de dois modos de operação do software:

- Modo de Aprendizado: Indicado pelo fluxo da numeração em azul da Figura 2. Neste modo, o usuário deve selecionar documentos de seu interesse que abordem um assunto específico, para compor o repositório local de documentos e apresentá-los ao aplicativo, para que haja o aprendizado do padrão conceitual de tal coleção. Os espaços de conceitos gerados podem ser armazenados em arquivos para usos futuros. Este modo deve ser executado para que o modo de filtragem possa funcionar.

- Modo de Filtragem: Indicado pelo fluxo da numeração em vermelho da Figura 2. Neste modo, o usuário poderá solicitar uma busca, a partir da definição de qual espaço de conceitos produzido no outro modo será considerado. Desta forma, são recuperados e filtrados os documentos, classificando-os de acordo com a relevância resultante da ativação da RNA de Hopfield.

Foi criado um *Product Backlog* que, neste caso, foi um reflexo dos módulos que compõem a arquitetura do software de forma enumerada e outros itens necessários para o trabalho. Já o diagrama *Product Burndown* foi criado de acordo com as estimativas do *Product Backlog* de forma que se tivesse a visão futura do andamento do desenvolvimento. Posteriormente neste capítulo, na Figura 3 e Figura 4, serão apresentadas duas versões deste diagrama: uma que representa o planejamento realizado para o desenvolvimento e a outra que contempla as informações efetivas do desenvolvimento para efeitos de comparação, respectivamente

O Quadro 1 a seguir apresenta o *Product Backlog* definido no início do desenvolvimento:

Quadro 1. *Product backlog* para o desenvolvimento.

Item	Estimativa (Dias)	Esforço e Risco	Valor
Rede Neural Artificial de Hopfield (RNA)	30	Alto	Alto
Módulo de Geração de Espaço de Conceitos (GEC)	20	Alto	Alto
Integração RNA – GDC	10	Médio-Alto	Alto
Módulo de Indexação Automática de Documentos (IAD)	20	Médio	Alto
Integração GDC – IAD	10	Médio-Baixo	Alto
Módulo de Obtenção de Documentos (MOD)	10	Médio-Baixo	Médio
Módulo de Persistência de Dados (MPD)	5	Baixo	Baixo
Módulo de Interatividade (MI)	10	Baixo	Baixo

Os itens de *backlog* do produto final foram distribuídos em quatro *Sprints* de acordo com sua prioridade. Os mesmos serão detalhados neste capítulo nas seções 4.1 a 4.4. Entretanto, para uma visão geral do desenvolvimento da aplicação, agora serão apresentados os diagramas de *burndown* inicial e final, que refletem um comparativo entre o planejado e o realizado. Os *Sprints* foram planejados inicialmente para apresentar uma diluição do esforço de trabalho, conforme a Figura 10.

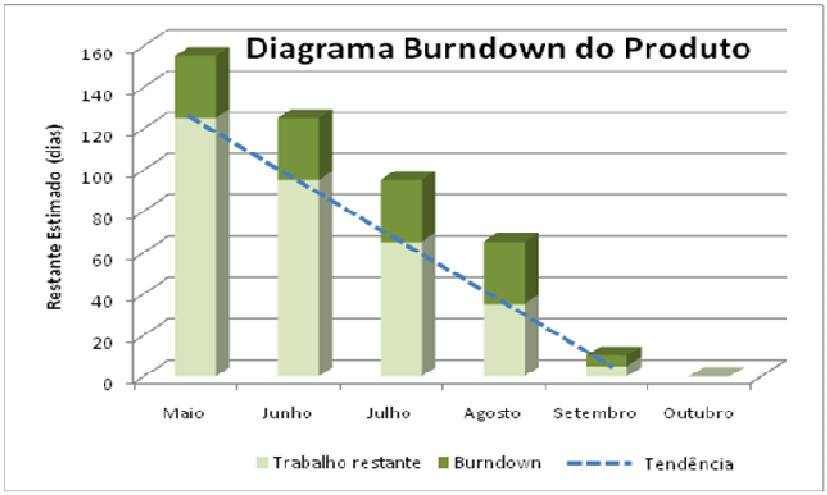

Figura 10. Diagrama de *Burndown* planejado para o trabalho.

O trabalho restante, indicado no diagrama de *Burndown,* representa a quantidade de dias de trabalho restantes estimados no *Product Backlog.* O *Burndown,* indica a quantidade de dias que foram trabalhados no desenvolvimento do projeto. Logo, a medida que o tempo passa, é subtraído do trabalho restante a quantidade de *Burndown.*

Após o desenvolvimento o diagrama final é o da Figura 11:

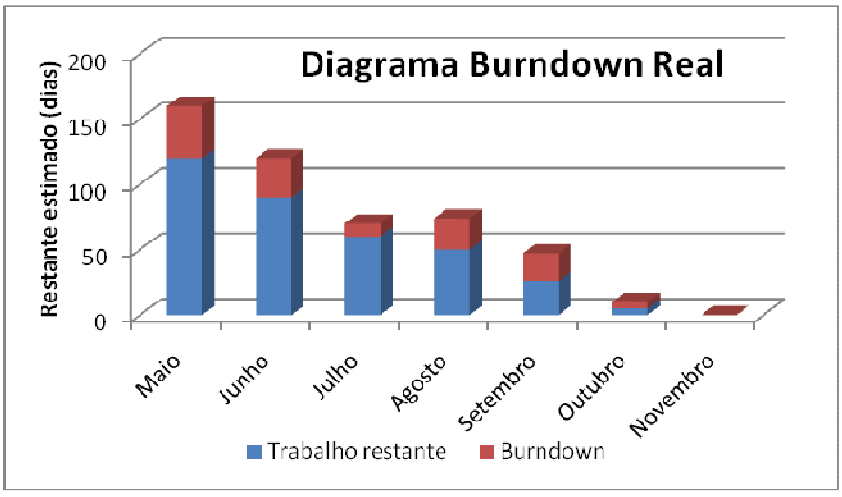

Figura 11. Diagrama de *Burndown* efetivamente realizado no trabalho.

A composição e desenvolvimento dos *Sprints*, bem como as razões pelas quais o *burndown* divergiu do planejado, são explicadas em detalhes a seguir.

4.1 Complexidade e Tecnologias Utilizadas

Os itens principais de complexidade foram a indexação automática de documentos, a geração de espaço de conceitos e a RNA de Hopfield, apresentados no capítulo de Referencial Teórico. A complexidade é justificada pelo inicial desconhecimento e conseqüentemente a necessidade do estudo árduo realizado dos algoritmos que os compõe. Houve um difícil processo, desde o entendimento desses itens, passando pela modelagem até a correta codificação dos algoritmos.

Outro item de complexidade foi o uso da Yahoo Search API, para prover a funcionalidade de busca em repositório de páginas *on line.* Por nunca ter sido usada antes, houve também a necessidade de pesquisa e entendimento dos procedimentos necessários para a integração com a aplicação desenvolvida neste trabalho.

A aplicação desenvolvida foi construída utilizando o paradigma de programação orientada a objetos, utilizando a linguagem de programação Java. Inicialmente foi utilizada a ferramenta de desenvolvimento Eclipse (ECLIPSE, 2007) e posteriormente, para a interface gráfica, o NetBeans (NETBEANS, 2007). Já para a modelagem e projeto das classes foram criados diagramas UML (Unified Modeling Language) através do *framework* Omondo (OMONDO, 2007).

4.1.1 Yahoo Search API

A Yahoo Search Application Programming Interface (API) (YAHOO, 2007) é um *framework* livre que permite aos desenvolvedores de diversas linguagens utilizarem os serviços de busca do *Yahoo* dentro de suas aplicações. Através dela, é possível incorporar nos softwares desenvolvidos a funcionalidade de pesquisa na *Web,* de forma que os resultados das buscas possam ser tratados por eles da maneira desejada.

Para tanto, é necessário um registro gratuito para utilização do serviço, através do qual o desenvolvedor obtém uma chave de acesso requerida no

momento da abertura da conexão com o provedor do serviço. Além disso, por questões de política de acesso, há também o limite diário de cinco mil requisições de buscas solicitadas via esta API, que porém é suficiente as necessidades deste trabalho.

4.1.2 HTML Parser

O HTML Parser (HTML, 2006) é uma biblioteca escrita em Java, usada para realizar o *parsing* de documentos HTML. Ela disponibiliza facilidades de transformação e extração de conteúdos de arquivos HTML.

Sua licença de uso está regida sob os termos da *Common Public License* (CPL) v 1.0, o que permitiu seu uso neste Trabalho de Conclusão de Curso. Seu pacote de distribuição foi concebido de maneira robusta, ou seja, para suportar diversos padrões de código HTML, e o mesmo é bem testado por sua comunidade de desenvolvimento, o que garante a qualidade e eficiência de seus métodos.

O HTML Parser foi usado neste trabalho na etapa de indexação automática, onde é necessária a eliminação de quaisquer *tags* HTML dos documentos. Desta forma, através do *parsing* disponibilizado por essa biblioteca, foi possível a extração do conteúdo texto dos documentos.

4.2 Sprint 1

Por se tratar do núcleo da solução do trabalho e também assunto de total desconhecimento inicial, a RNA de Hopfield teve a maior prioridade e por isso o desenvolvimento do software se iniciou por ela, sendo assim selecionada para o primeiro S*print.*

Uma proposta levantada junto ao Prof. Co-Orientador foi a de se utilizar uma RNA pronta de forma que a complexidade do trabalho poderia recair sobre outras funcionalidades. Então, foi realizada uma pesquisa em busca de soluções

já existentes que fossem implementadas na linguagem utilizada. Um ambiente de desenvolvimento interessante encontrado foi o Joone (MARRONE, 2007), que provê meios para a geração automática de redes neurais artificiais em Java.

Porém, a decisão tomada foi a de pesquisa e implementação própria por parte do aluno, uma vez que o uso de soluções prontas ou ferramentas como o Joone demandariam muito tempo para o seu aprendizado, além da necessidade de entendimento de suas interfaces para integração com os módulos a serem desenvolvidos. Em virtude do desenvolvimento próprio, a ferramenta *Sprint backlog* da metodologia foi utilizada para que fossem definidos os itens necessários para o desenvolvimento do *Sprint*. A definição das estimativas de horas para cada um dos itens foi feita de maneira a dedicar mais tempo ao estudo e entendimento da RNA de Hopfield.

O Quadro 2 a seguir apresenta o primeiro *Sprint backlog:*

Quadro 2. *Sprint backlog* para o S*print* 1.

Item	Estimativa (Horas)	Comentários
Leitura/Pesquisa do Algoritmo da RNA	50	Entendimento do mecanismo, pesquisa de soluções existentes.
Modelagem	10	-
Codificação	40	Implementação do código da RNA.
Testes	10	Comparação com o funcionamento obtido com o esperado.
Documentação	5	

O trabalho de MARIN (2003) foi a fonte de pesquisa inicial sobre o algoritmo da RNA de Hopfield, porém outros trabalhos foram consultados, como o de HEATON (2007) que já traz exemplos práticos de implementações. A partir destas pesquisas e da compreensão da estrutura de uma RNA de Hopfield, foi então possível a modelagem das classes. A Figura 12 mostra o diagrama de classes UML da mesma.

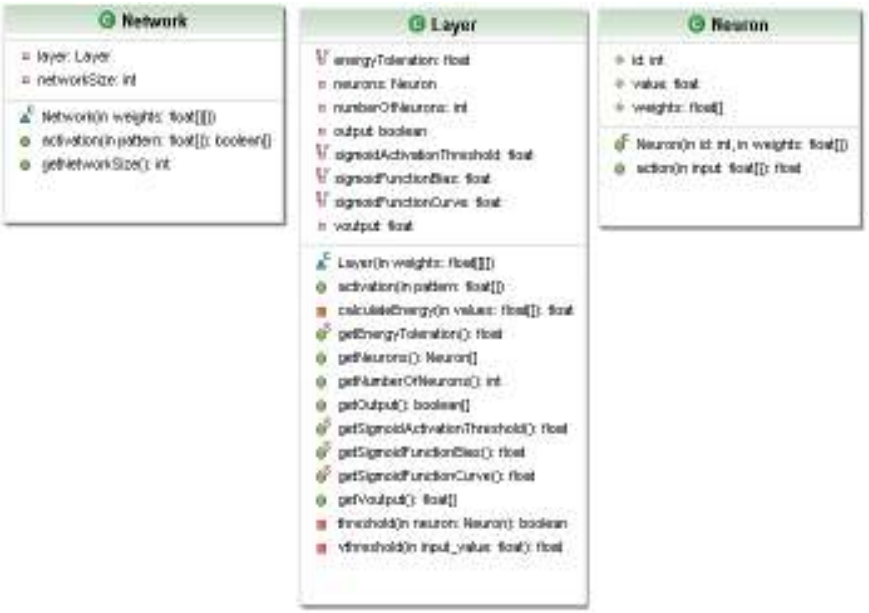

Figura 12. Diagrama de classes da RNA de Hopfield.

Este diagrama mostra que a RNA é composta pela classe Neuron, representando os neurônios que contém os métodos para a ativação; pela classe Layer, que representa a camada única da RNA de Hopfield, controlando todo o funcionamento e instanciação dos neurônios; e a classe Network que encapsula a classe Layer, para interface com os outros módulos. Como resultado, a RNA de Hopfield desenvolvida apresenta as seguintes características: função de ativação do tipo sigmóide; possibilidade de parametrização dos valores de viés, constante de curvatura da função de ativação, valor de patamar para ativação do neurônio e a energia para convergência; instanciação dinâmica do número de neurônios a partir das dimensões da matriz de pesos passada como parâmetro no construtor do objeto da classe.

Outra característica é que pelo fato da entrada da matriz de pesos da RNA ser gerada pelo módulo de geração de espaço de conceitos, sua matriz pode possuir valores assimétricos para pesos, uma vez que cada linha da matriz representa os valores dos pesos de entrada para os respectivos neurônios. Para o teste da RNA, utilizou-se como parâmetro o trabalho existente de HEATON (2007) o qual apresenta uma aplicação *applet* Java que implementa uma RNA de Hopfield genérica.

Para tanto, utilizou-se de redes compostas por quatro neurônios, como foi feito no trabalho de HEATON (2007) e realizou-se o treinamento das mesmas com o padrão [1,0,1,0]. Isso foi feito a fim de avaliar se suas saídas contemplam o caráter associativo esperado. O Quadro 3 mostra os valores de saídas para algumas entradas possíveis considerando o padrão armazenado mencionado.

Quadro 3. Valores de saída da rede para alguns padrões de entrada.

Padrão de entrada	Saída da rede
[0, 0, 0, 1]	[0, 1, 0, 1]
[0, 1, 0, 0]	[0, 1, 0, 1]
[0, 1, 1, 1]	[0, 1, 0, 1]

Esta análise, dirigida especificamente na atividade de *Sprint review*, monstrou o atendimento da expectativa de funcionamento da rede neural artificial, que é a associação e tolerância a distorções, sendo que se pôde notar a estabilização da rede sob o padrão armazenado quando possível à rede.

Desta forma, a característica associativa da rede é usada no momento da filtragem, onde são apresentados à rede para cada documento, os termos encontrados nos mesmos. Estes termos encontrados representam o padrão de entrada semelhante ao exemplo do Quadro 3, onde a ocorrência do termo resulta no valor 1 e do contrário 0 no vetor de entrada. A partir daí a rede pode ser ativada para avaliar se o documento é relevante ou não. A idéia é associar o número de neurônios ativos na saída com a relevância do documento, ou seja, quanto maior o número de neurônios ativos maior é a relevância do documento.

De maneira geral, durante a implementação foram encontradas dificuldades em virtude do desconhecimento do assunto, o que teve como conseqüência uma demanda maior de tempo de pesquisa sobre o mesmo. Isso fez com que ocorressem readequações das atividades planejadas, de forma que foi alocada maior dedicação de tempo ao *Sprint* para o término. Logo, ao final deste *Sprint,* não ficaram itens pendentes para serem alocados nos próximos desenvolvimentos.

4.3 Sprint 2

Para a segunda etapa de desenvolvimento, foi realizada inicialmente a construção do módulo de geração de espaço de conceitos (GEC), pois o mesmo foi o segundo item mais prioritário no *Product Backlog.* Segundo o planejamento, o seu desenvolvimento deveria ter se iniciado no período das férias escolares, o que não foi possível ser feito em sua totalidade, de maneira que apenas o estudo preliminar do algoritmo de geração de espaço de conceitos foi efetuado.

Para o atendimento dos prazos, a flexibilidade da metodologia permitiu a agregação de mais itens de *backlog* para este *Sprint.* Este item foi o estudo e pesquisa sobre o módulo de indexação automática de documentos (IAD) que era o próximo item no *Product backlog.* Logo, nesta fase houve a alteração da visão do diagrama de *burndown,* pelo motivo da inclusão deste item não previsto. A alteração ocorreu pelo fato do aumento da quantidade de trabalho alocada para o *Sprint,* em relação ao que se havia planejado. Logo, o diagrama transferiu o trabalho restante referente ao item adicionado, para o período de execução do *Sprint* 2.

Os Quadros 4 e 5 apresentam o segundo *Sprint backlog,* respectivamente referentes aos itens do módulo de geração de espaço de conceitos e o estudo do algoritmo do módulo de indexação automática de documentos, que foi adicionado para a compensação do não desenvolvimento mencionado nas férias escolares.

Quadro 4. *Sprint backlog* referente ao módulo de geração de espaço de conceitos para o *Sprint* 2.

Item	Estimativa (Horas)	Comentários
Algoritmo	10	Análise e entendimento do algoritmo relacionado.
Modelagem	3	
Código	40	Seleção dos melhores termos;cálculos relativos às ocorrências dos termos; cálculos referentes aos pares de temos; cálculos dos coeficientes assimétricos de similaridade.
Teste	20	Integração com a RNA.
Documentação	5	

Quadro 5. *Sprint backlog* do módulo de indexação automática de documentos para o *Sprint* 2.

Item	Estimativa (Horas)	Comentários
Algoritmo	20	Entendimento, pesquisa de algoritmos de lematização; busca de alternativas para parsing dos documentos.

Com relação à geração de espaço de conceitos, o resultado foi fruto de estudo e pesquisa. A dificuldade encontrada foi o esclarecimento dos passos necessários, uma vez que o algoritmo é compreendido por uma série de cálculos para a produção de espaço de conceitos e por isso neste item se gastou mais tempo do que o estimado. O resultado da investigação dos procedimentos para o módulo de geração de espaço de conceitos é o diagrama de classes da Figura 13:

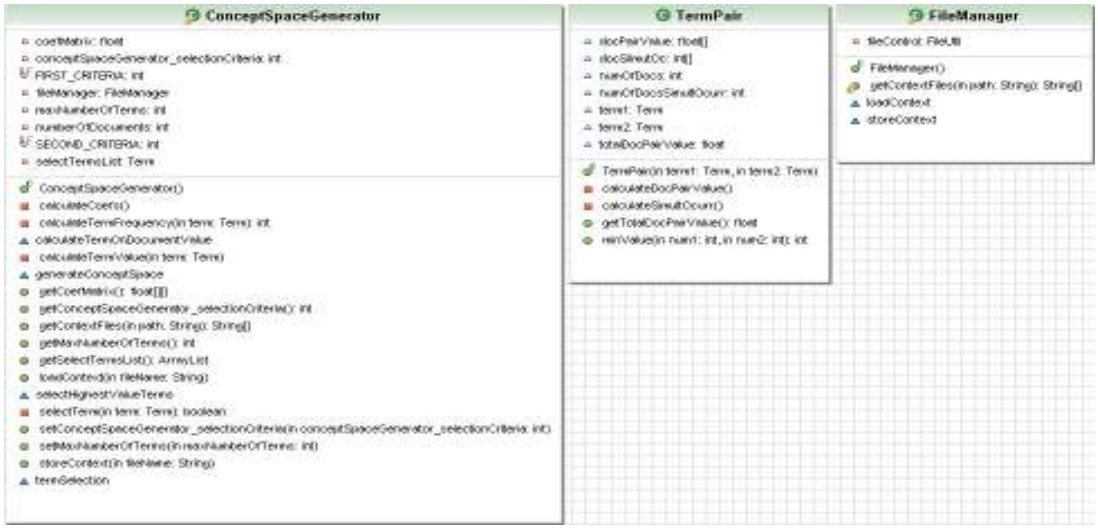

Figura 13. Diagrama de classes do módulo de geração de espaço de conceitos.

Este módulo de geração de espaço de conceitos, foi composto pela classe principal ConceptSpaceGenerator, que contém todos os métodos e procedimentos para os cálculos do algoritmo apresentados na seção 2.1 do Referencial Teórico; pela classe TermPair, que representa um par de termos, ou seja, encapsula dois objetos da classe Term e o FileManager para o gerenciamento dos conceitos armazenados em arquivo do tipo *Text File* (txt). Como resultado, o módulo de geração de espaço de conceitos desenvolvido ficou com a seguinte configuração: parametrização do critério de seleção dos termos,

que podem ser dois: um baseado na freqüência dos termos e o segundo pela importância do termo em relação à coleção de documentos.

Há também a possibilidade de: parametrização do número máximo de termos a serem selecionados no processo de geração dos espaços de conceitos, uma vez que o número de termos selecionados (que resulta no número de neurônios da RNA) deve ser configurável para se ter um maior controle sobre o comportamento da RNA. Além disso, há a necessidade da persistência dos espaços de conceitos gerados em arquivos no formato txt. Vale ressaltar aqui que a funcionalidade de armazenamento não foi desenvolvida neste momento, apenas modelada. A mesma foi realmente implementada no *Sprint 4,* onde o módulo de persistência de dados foi criado. Ela é utilizada para o armazenamento de espaços de conceitos para o seu uso em filtragens posteriores.

Encontraram-se dificuldades no teste deste módulo, pois nos testes iniciais os valores apresentados na saída do módulo de geração de espaço de conceitos, não respeitavam a faixa de valores reais entre 0 e 1. Esta faixa deve ser respeitada, uma vez que seus valores representam probabilidades, onde os valor 0 equivale a 0% e o valor 1 a 100%. Pelo fato deste módulo ser composto por diversos cálculos co-relacionados, revisões da implementação do algoritmo foram feitas a fim de se identificar as discordâncias com o algoritmo de geração de espaço de conceitos.

Em esclarecimentos realizados junto ao Co-Orientador deste trabalho, pôde-se resolver o problema, uma vez que existiam partes obscuras para o aluno sobre o algoritmo proposto. Os equívocos disseram respeito à fórmula de cálculo do valor dos pares de termos em relação aos documentos, o que impactava na função de agrupamento assimétrica, que é responsável pela saída final dos valores do módulo. A avaliação e resolução destes problemas foram feitas na atividade de revisão do *Sprint,* não relatada.

A respeito da parte correspondente ao desenvolvimento do módulo de indexação automática de documentos neste *Sprint*, foi feito o estudo do mesmo. Desta forma, foram definidos os escopos pertinentes para este módulo, bem como a pesquisa para implementar seus requisitos.

Neste momento, foi buscada uma solução para a funcionalidade de lematização dos termos, pois era um ponto de desconhecimento. Logo, com a indicação do Co-Orientador, encontrou-se uma implementação de lematização e remoção de *stopwords* feita na mesma linguagem utilizada, desenvolvida por DIAS (2004). Finalmente, estas implementações foram analisadas e planejadas para serem aproveitadas e incluídas no próximo *Sprint,* quando o módulo de indexação automática de documentos foi efetivamente implementado.

4.4 Sprint 3

Dando continuidade ao último item iniciado no desenvolvimento do *Sprint* anterior, nesta etapa, foi feita a implementação do módulo de indexação automática. Como o estudo sobre seu funcionamento já tinha sido feito anteriormente, o projeto das classes já estava praticamente concluído. Em contrapartida, com a alteração do *Sprint backlog* do *Sprint 2,* para manter a evolução do desenvolvimento com a previsão de término nos prazos estipulados de acordo com o *burndown*, o próximo item do *Product Backlog* teve que ser incluído neste *Sprint* para uma maior diluição do trabalho restante.

O item incluído foi o módulo de obtenção de documentos (MOD). Portanto, o *Sprint backlog* deste *Sprint* contempla os itens de ambos os módulos conforme os Quadros 6 e 7 a seguir:

Quadro 6. *Sprint backlog* do módulo de indexação automática de documentos para o *Sprint 3*.

Item	Estimativa (Horas)	Comentários
Modelagem	3	Definição dos objetos que representaram as entidades descritas no algoritmo; definição de premissas para integração com os outros módulos.
Código	20	Parsing de documentos HTML; identificação e remoção de stopwords; lematização das palavras; formação dos termos.
Teste	10	Integração com os módulos.
Documentação	5	

Quadro 7. *Sprint backlog* do módulo de obtenção de documentos para o *Sprint* 3.

Item	Estimativa (Horas)	Comentários
Modelagem	1	Definição dos objetos necessários.
Código	25	Interface com Yahoo Search API e busca local.
Teste	15	Integração com os módulos.
Documentação	5	

A investigação dos componentes necessários para o módulo de indexação automática resultou no diagrama de classes da Figura 14 abaixo:

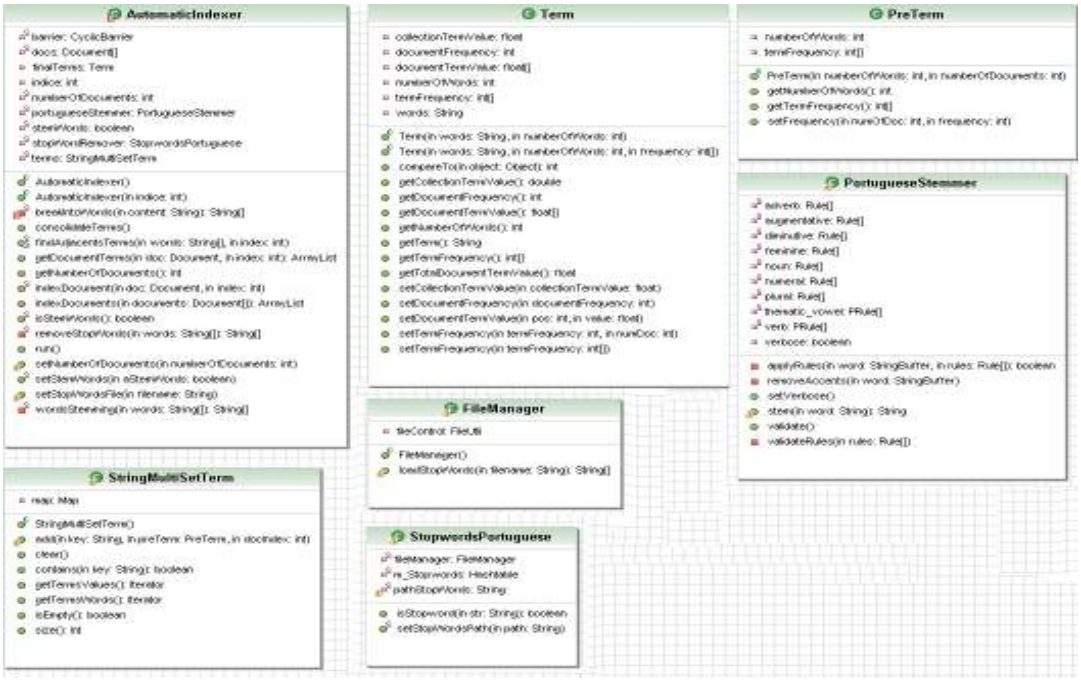

Figura 14. Diagrama de classes do módulo de indexação automática.

As classes compreendem a AutomaticIndexer, que é a classe principal responsável pela condução do fluxo do algoritmo de indexação automática de documentos; a Term que representa um termo encontrado em um documento; a PreTerm que é uma inicialização prévia dos termos utilizada pela classe StringMultiSetTerm, que calcula a freqüência dos termos nos documentos; as classes PortugueseStemmer e StopwordsPortuguese, que compreendem as implementações de lematização e remoção de *stopwords* obtidas e adaptas a partir do trabalho de DIAS (2004); e a FileManager, responsável por gerenciar os arquivos de *stopwords* em arquivos do tipo .txt.

Os passos de indexação automática são realizados por *threads.* Para cada documento a ser indexado é criada uma *thread* que faz o processamento do documento. Isso foi desenvolvido para uma otimização de funcionamento do módulo. Outro item de otimização é o modo do cálculo da freqüência dos termos, que é realizado através do uso de um mapa de *hash,* disponível na linguagem Java. Essa foi uma alteração feita durante o desenvolvimento, uma vez que se sabe que esta estrutura de dados é muito eficiente para conjunto de dados indexados, que é o caso da contagem das freqüências dos termos nos documentos.

É possível a parametrização, feita via tela disponível na aplicação, do uso da funcionalidade de lematização de palavras segundo as regras para a língua portuguesa e a definição do conjunto de *stopwords* a ser considerado através da leitura de arquivos no formato txt. Novamente, vale lembrar que a funcionalidade de persistência apenas foi modela neste *Sprint,* pois a mesma foi efetivamente implementada no *Sprint 4* no módulo de persistência de dados.

Com relação ao desenvolvimento do módulo de obtenção de documentos, surgiram dificuldades no momento da utilização da Yahoo Search API e do HTML Parser, uma vez que os mesmos nunca tinham sido utilizados pelo aluno. Porém, graças à boa documentação disponibilizada dessas ferramentas, após o entendimento dos parâmetros para acesso ao serviço da API, bem como o entendimento das funcionalidades necessárias do HTML Parser para a extração de conteúdo do texto dos arquivos HTML e a leitura de *tags,* foi possível proceder com o desenvolvimento. Além disso, também houve a implementação da busca de documentos em repositório local.

Para este módulo, não se considerou vantajosa a elaboração de diagramas de classes, pois suas funcionalidades estavam bem claras e restritas ao uso das ferramentas mencionadas. Após o seu desenvolvimento, o módulo ficou com as seguintes características: busca de arquivos apenas do tipo HTML; e a limitação do número de resultados por página da busca via Yahoo Search API de 50 documentos, que é o número estabelecido pelo serviço e flexibilidade para também operar com repositório local de documentos.

Portanto, o módulo de obtenção de documentos pôde atender à necessidade da aplicação de obter documentos a partir de páginas *Web* locais ou remotas para a alimentação do módulo de indexação automática. Nos testes realizados, na busca documentos e a realização de *parsing* HTML, até a execução dos procedimentos de indexação automática, ambos os módulos desenvolvidos apresentaram um bom desempenho no trabalho em conjunto. Fato este que se deu graças à robustez da biblioteca HTML Parser e das soluções adotadas para implementação do módulo de indexação automática para ganho de desempenho.

4.5 Sprint 4

Na última etapa de desenvolvimento, foram codificados os módulos de persistência de dados e o de interatividade (MI). Este último é responsável pela interface gráfica da aplicação. Os Quadros 8 e 9 mostram o *Sprint backlog* para o *Sprint* 4.

Quadro 8. *Sprint backlog* referente ao módulo de persistência de dados para o *Sprint* 4.

Item	Estimativa (Horas)	Comentários
Modelagem	2	
Código	15	Dicionário de *stopwords* para o módulo de Indexação Automática; persistência para os espaço de conceitos gerados no módulo de Geração de Espaço de Conceitos.
Teste	3	Integração com o módulo de Geração de Espaço de Conceitos; integração com o módulo de Indexação Automática.
Documentação	5	

Quadro 9. *Sprint backlog* referente ao módulo de interatividade para o *Sprint* 4.

Item	Estimativa (Horas)	Comentários
Modelagem	10	Definição do *layout*, menus e opções necessárias.
Código	30	Criação de *Front-End* e interação com as funcionalidades criadas.
Teste	15	Teste geral da interface.
Documentação	3	

O módulo de persistência de dados foi construído atendendo à necessidade dos módulos de indexação automática e de geração de espaços de conceitos para a gravação e leitura de seus dados em arquivos no formato txt. Já a respeito da interface gráfica, desenvolvida com NetBeans (NETBEANS, 2007), a mesma teve apenas o papel de disponibilizar ao usuário da aplicação o acesso a todas as funcionalidades já descritas nesse desenvolvimento.

Como previsto pela metodologia, neste último *Sprint* foram implementadas as funcionalidades menos complexas. Isso propiciou a não alteração dos prazos estabelecidos.

Com relação às funcionalidades implementadas, a Figura 8 mostra um exemplo de arquivo txt gerado para a persistência dos espaços de conceitos, que representa uma matriz de coeficientes assimétricos de similaridade para uma coleção de documentos utilizada como exemplo:

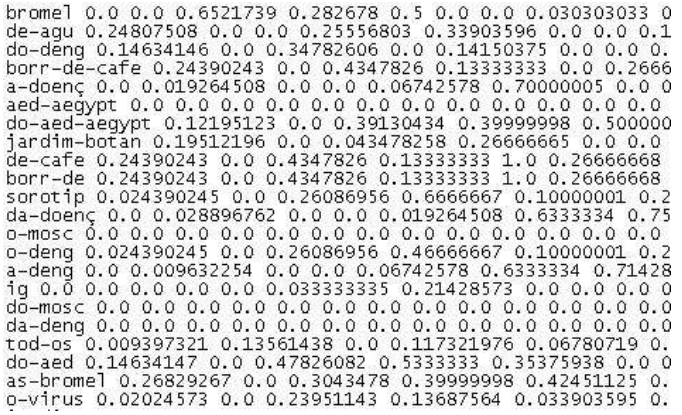

Figura 15. Arquivo texto de um espaço de conceito gerado pelo módulo de persistência de dados.

Cada linha de entrada deste arquivo gerado representa uma linha da matriz de coeficientes mencionada. Porém, no início de cada linha é armazenado o termo correspondente à entrada.

Já sobre a interface gráfica, os resultados são apresentados na próxima seção 4.6, onde todas as telas da aplicação são apresentadas.

4.6 Telas da Aplicação

São apresentadas a seguir, as telas da aplicação desenvolvida como resultado deste Trabalho de Conclusão de Curso.

Figura 16. Tela inicial a aplicação desenvolvida.

O menu de opções, mostrado na Figura 16, possibilita o acesso às configurações da aplicação, o acesso ao modo de aprendizado e ao modo de filtragem.

Já a Figura 17, mostra a tela disponível ao se escolher no menu o item de Configurações e em seguida o sub item Parâmetros, onde é possível realizar acessar as configurações de funcionamento da indexação automática de documentos, da geração de espaço de conceitos e da RNA de Hopfield.

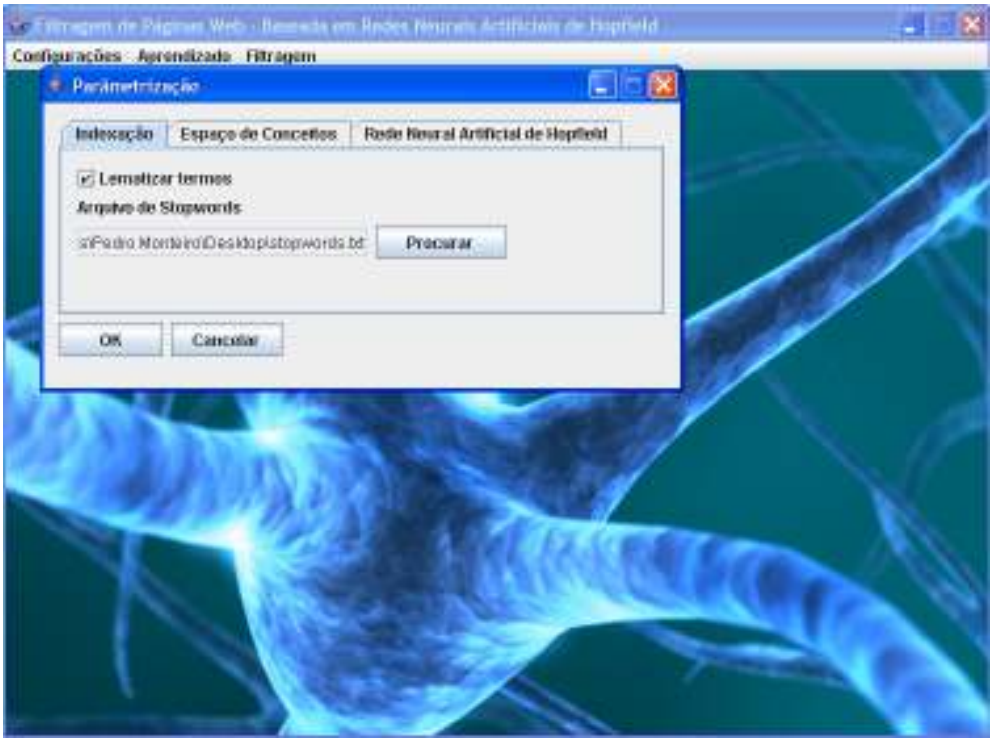

Figura 17. Tela de parametrização da indexação automática de documentos.

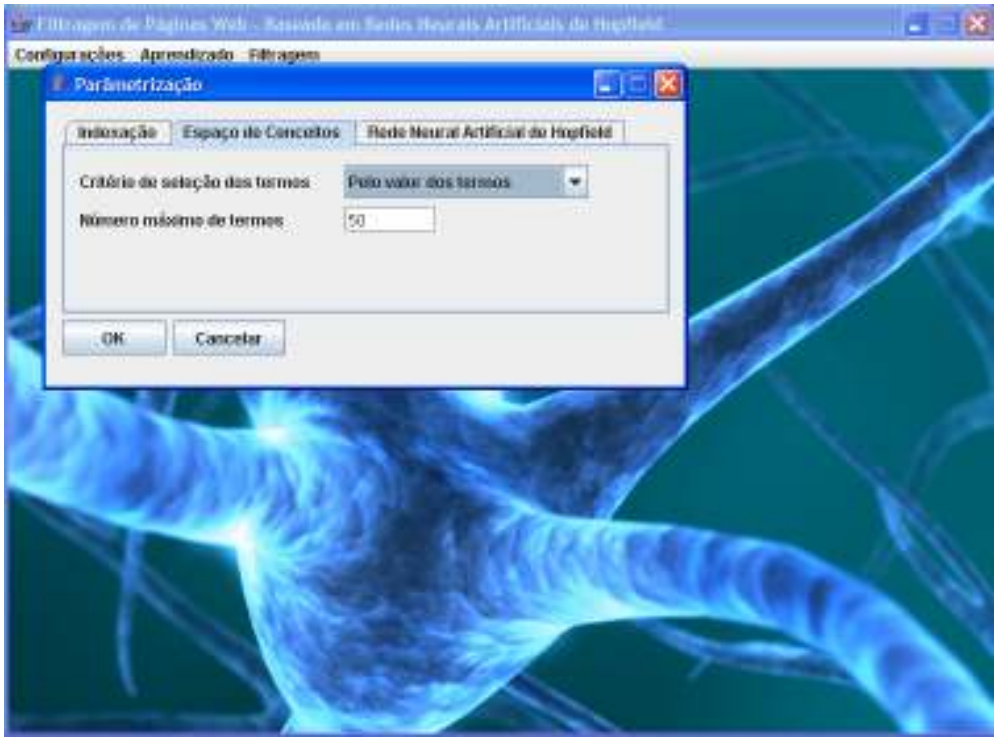

Figura 18. Tela de parametrização da geração de espaço de conceitos.

A Figura 18 mostra a mesma tela da Figura 17, porém na aba que é disponibilizada a configuração da geração de espaço de conceitos. Já a Figura 19, mostra a tela de configuração de operação da RNA de Hopfield.

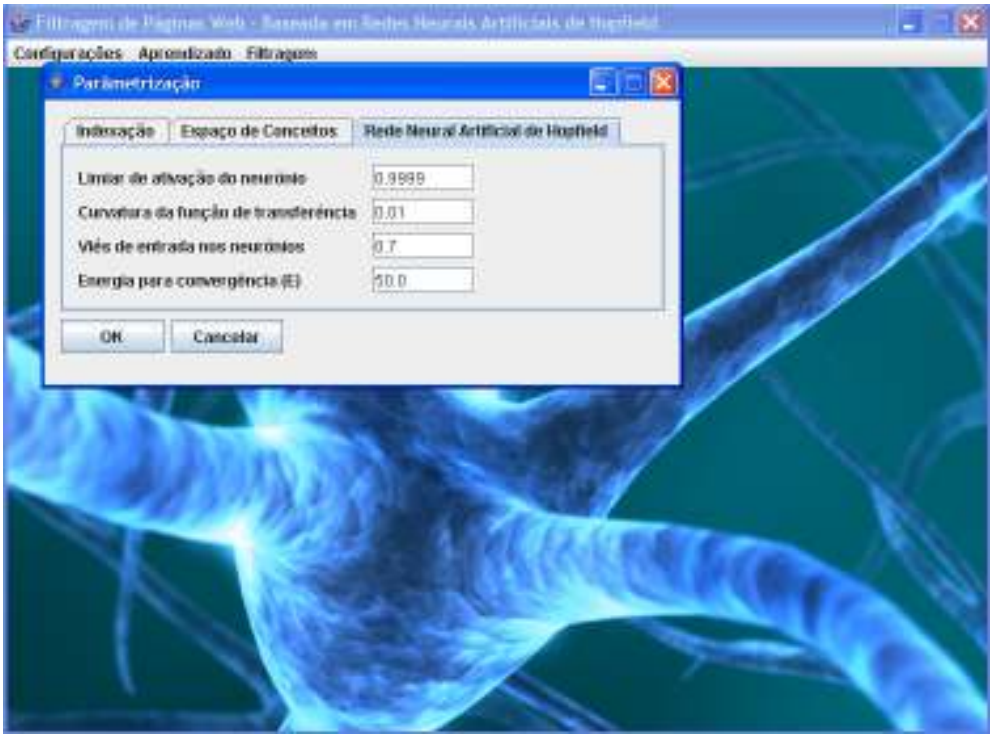

Figura 19. Tela de parametrização da RNA de Hopfield.

A Figura 20 mostra a mesma tela disponível ao acessar a opção de Aprendizado do menu principal. Nesta tela, pode ser feita a geração de espaço de conceitos a partir de uma coleção de documentos indicada num diretório local. Além disso, o local de armazenamento dos arquivos de espaços de conceitos pode ser configurado. É possível também, carregar quaisquer conceitos armazenados no diretório configurado, para que os mesmos sejam considerados nas filtragens, mostrado na Figura 21.

Figura 20. Tela de geração de espaços de conceitos.

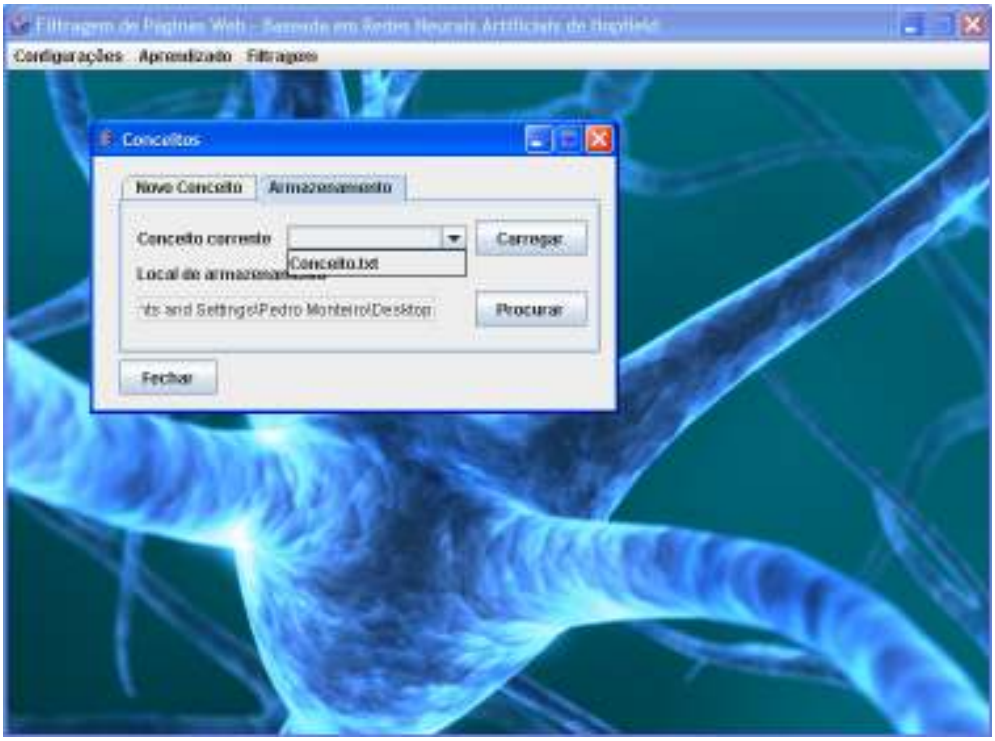

Figura 21. Tela de armazenamento de espaços de conceitos.

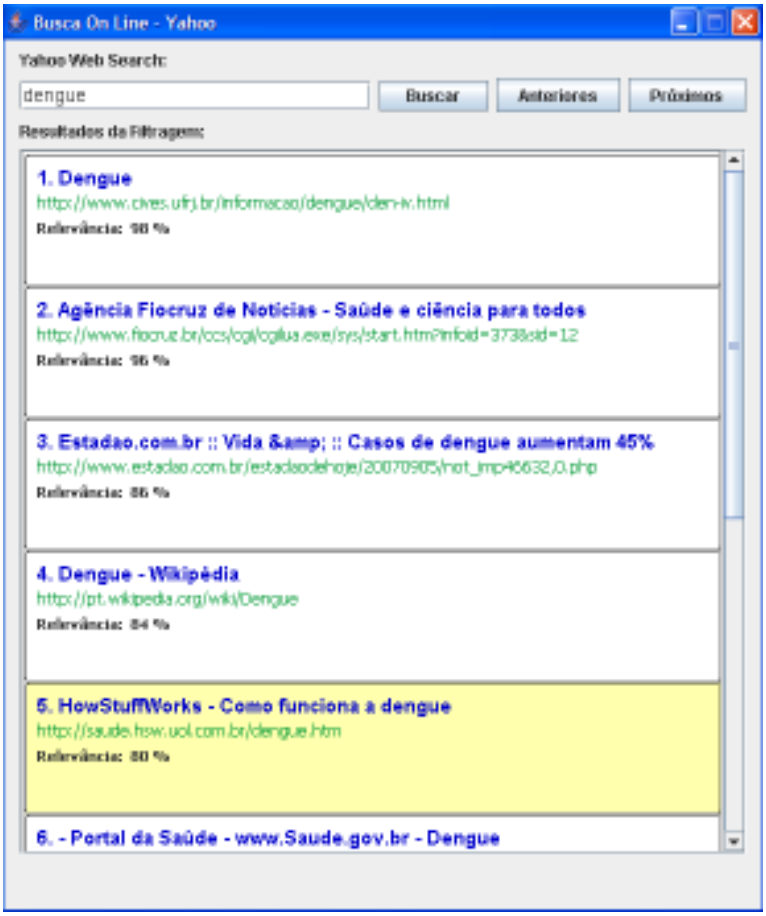

Figura 22. Tela de filtragem da aplicação.

A Figura 22 mostra a tela de filtragem da aplicação, que permite busca *on line* de páginas do repositório acessado através da Yahoo Search API. Após a análise da relevância de cada documento, os mesmos são ordenados de acordo na listagem.

Para cada documento, são exibidas as informações de título (fornecido pela API), *link* e a porcentagem de relevância do documento em relação ao espaço de conceito considerado na filtragem. Ao clicar sobre um documento listado, o mesmo é aberto no navegador padrão da máquina do usuário para a visualização de seu conteúdo.

5 AVALIAÇÃO E VALIDAÇÃO

Remetendo ao objetivo definido para este trabalho, para avaliação e validação do mesmo, as taxas de Precisão e de Cobertura são calculadas da seguinte forma:

Taxa de Precisão (P) = número de documentos relevantes recuperados / número total de documentos recuperados.

Taxa de Cobertura (C) = número de documentos relevantes recuperados / número de documentos relevantes.

Para avaliação e validação do trabalho, deve ser utilizada pelo aluno, uma coleção de documentos satisfatoriamente relevante. Logo, foi determinado o uso do repositório público disponibilizado pela Linguateca (LINGUATECA, 2007). A Linguateca é um centro de recursos para o processamento computacional da língua portuguesa, que provê informações e modelos de avaliações para sistemas de RI.

O nome da coleção a ser usada, disponível na Linguateca, é a Folha-RIcol. Essa coleção de documentos é derivada do *corpus* do NILC[5] (Núcleo Interinstitucional de Lingüística Computacional), que contém artigos provenientes do Jornal Folha de São Paulo do ano de 1994. A coleção completa possui 5090 artigos, com a classificação dos documentos relevantes para 18 assuntos.

Dentre os assuntos presentes nesta coleção, deve ser usado o assunto Música Brasileira para a realização de experimentos de filtragem. O assunto contém no total 69 documentos. Para documentos não relevantes, devem ser utilizados outros 10 documentos que abordem assuntos totalmente diferentes, como Futebol, Cinema e Economia.

No entanto, dentre os 69 documentos relevantes, devem ser considerados apenas 15 documentos, escolhidos aleatoriamente.

[5] http://www.nilc.icmc.usp.br/nilc/

Isso se deve ao fato de que geralmente, na avaliação de sistemas de RI, as taxas de Precisão e de Cobertura são medidas na presença de uma quantidade de documentos relevantes que seja de mesma grandeza da quantidade de documentos não relevantes.

Logo, o universo de busca a ser utilizado no modo de filtragem, deve ser composto por 25 documentos (15 documentos relevantes e os 10 documentos indicados como não relevantes).

Para a geração dos espaços de conceitos, no modo de aprendizado, devem ser utilizados outros 20 documentos aleatórios relevantes da coleção. Essa definição é assumida, na medida em que é bem mais plausível para a avaliação, que o número de documentos apresentados como exemplo, seja menor do que o número de documentos do universo de busca da filtragem.

Semelhante à avaliação conduzida no trabalho de MARIN (2003), para o cálculo das taxas de Precisão e de Cobertura, devem ser considerados 11 patamares de porcentagem (5%, 10%, 20%, 30% 40%, 50%, 60%, 70%, 80%, 90% e 100%) das relevâncias resultantes das filtragens, indicadas para os documentos. Esses patamares devem indicar o piso de relevância apontada nos documentos para os cálculos das taxas de Precisão e Cobertura. Logo, nos experimentos, deve ser medido o número de documentos relevantes recuperados e o número total de documentos recuperados para cada um dos patamares.

Para avaliar o alcance do objetivo, deve-se verificar se em algum dos experimentos realizados há um cenário em que uma filtragem tenha sido feita com altos patamares de relevância (a partir de 80%) e tenham sido atingidas taxas de Precisão e Cobertura de mesma grandeza (em torno de 70% a 90%).

6 CONCLUSÃO

Este trabalho é motivado pelo crescimento espantoso e ininterrupto da *Web* que tem como conseqüência a existência de um acervo de informações tão grande que dificulta a tarefa de recuperação de informação da rede por seus usuários. Neste cenário, a aplicação desenvolvida tratou do problema da eficácia na filtragem de páginas *Web* através de uma abordagem que se baseia em RNA de Hopfield a fim de prover resultados de buscas mais apropriados aos interesses dos usuários.

Para sua realização, foi necessária a combinação de técnicas da área da Recuperação da Informação e Inteligência Artificial para a criação de um modelo plausível para a resolução do problema. Estas técnicas foram a indexação automática de documentos, geração de espaço de conceitos que proveram a alimentação para uma RNA de Hopfield.

O uso da metodologia *Scrum,* que prioriza os itens mais críticos do projeto para serem desenvolvidos no começo, propiciou um progresso mais confortável para a implementação dos requisitos do sistema, uma vez que grande parte do mesmo contemplou algoritmos e conceitos desconhecidos inicialmente pelo aluno. Além disso, o seu acompanhamento de evolução do software e flexibilidade para a realocação dos itens de *backlog* no cronograma, nas etapas de desenvolvimento, foram fatores cruciais para a produção do trabalho.

As principais dificuldades foram encontradas na implementação dos algoritmos de RI, que compreendem na indexação automática de documentos e na geração de espaço de conceitos. Com relação à RNA de Hopfield, foi necessário um grande estudo para a modelagem da solução final obtida.

Apesar das limitações que dizem respeito aos desafios intrínsecos de processamento de língua natural e da diversidade de tipos de conteúdos na *Web,* o trabalho propiciou uma rica experiência na área RI.

Para encaminhamento de trabalhos futuros, podem ser adotadas mais técnicas para o aumento da eficácia das filtragens, como as ontologias. Elas seriam utilizadas para uma melhor representação dos conceitos das palavras, de maneira que os termos possam ser identificados de um modo mais abrangente, através de atributos que os generalizem, para a consideração de outras

possibilidades dentro do contexto analisado. Logo, não seriam apenas considerados os termos encontrados nos documentos, mas também outros termos generalizados a partir das ontologias.

Além disso, outras melhorias podem ser feitas com a agregação de tratamento de outros tipos de documentos além do tipo HTML existentes para uma melhor abrangência do conteúdo da *Web.* Outra idéia é o uso do trabalho como um agente inteligente, que de forma transparente ao usuário, buscaria da *Web,* páginas que fossem julgadas relevantes e as apresentaria ao usuário.

7 REFERÊNCIAS

BEEDLE, M. et. al. Manifesto for Agile Software Development, 2001. Disponível em: <http://agilemanifesto.org/principles.html>. Acesso em 25 mar. 2007.

CHEN, H.; HSU, P.; ORWIG, R.; HOOPES, I.; NUNAMAKER, J. F. *Automatic Concept Classification of Text from Electronic Meetings*. Communications of the ACM, 37(10):56-73, 1994.

CHUNG, Y.; POTTENGER, W. M.; SCHATZ, B. R. *Automatic Subject Indexing Using as Associative Neural Network*. In: 3RD INTERNATIONAL ACM CONFERENCE ON DIGITAL LIBRARIES. Pittsburgh, Pa. Jun, 1998, 59-68.

CONCHANGO. Scrum For Team System - Process Guidance, 2006. Disponível em <http://www.scrumforteamsystem.com/en/default.aspx>. Acesso em 27 mar. 2007.

DIAS, M. A. L. *Extração Automática de Palavras-Chave na Língua Portuguesa Aplicada a Dissertações e Teses da Área das Engenharias.* 2004. Dissertação de Mestrado. FEEC-UNICAMP, Campinas, 2004

ECLIPSE. Eclipse Project, 2007. Disponível em: < http://www.eclipse.org/>. Acesso em 01 jun. 2007.

HEATON, J. Java Neural Networks, 2007. Disponível em: <http://www.jeffheaton.com/ai>. Acesso em 05 jun. 2007.

HTML Parser, 2006. Disponível em: <http://htmlparser.sourceforge.net/>. Acesso em 20 set. 2007.

LINGUATECA, 2007. Disponível em: <http://www.linguateca.pt/>. Acesso em 07 dez. 2007.

MARIN, A.. *Um Mecanismo para Filtragem de Páginas da Web baseado no modelo de Rede Neural Artificial de Hopfield.* 2003. 91 f. Dissertação (Mestrado em Informática). Centro de Ciências Exatas, Ambientais e de Tecnologias. Pontifícia Universidade Católica de Campinas, Campinas, 2003

MARRONE, P. Joone - Java Object Oriented Neural Engine, 2007. Disponível em: < http://www.jooneworld.com/>. Acesso em 01 jun. 2007.

MOKABYTE 85 - Processi e metodologie di sviluppo - II parte: le metodologie agili, 2004. Disponível em: <http://www.mokabyte.it/2004/05/metod-2.htm>. Acesso em 25 ago. 2007.

NETBEANS, 2007. Disponível em: < http://www.netbeans.org/index.html>. Acesso em 02 set. 2007.

NETO, H. P.; VIERIA, S. R. T. *Implementação de um mecanismo de Recuperação de Informação na web baseado em redes neurais.* 2005. Monografia (Bacharelado em Ciência da Computação). Faculdade Ruy Barbosa, Salvador, 2005.

OMONDO The Live UML Company, 2007. Disponível em: < http://www.eclipsedownload.com>. Acesso em 10 mai. 2007.

SCHWABER, K. Advanced Development Methods. SCRUM Development Process, 2006. Disponível em: <http://jeffsutherland.com/oopsla/schwapub.pdf>. Acesso em 20 ago. 2007.

YAHOO, Yahoo! Search SDK Software, 2007. Disponível em: <http://developer.yahoo.com/download>. Acesso em 15 set. 2007.