UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Daniel Neis Araujo

INTEGRANDO APRENDIZAGEM DE MÁQUINA AO MOODLE PARA MELHORIA DOS FÓRUNS DE DISCUSSÃO

Florianópolis (SC)

2014

Daniel Neis Araujo

INTEGRANDO APRENDIZAGEM DE MÁQUINA AO MOODLE PARA MELHORIA DOS FÓRUNS DE DISCUSSÃO

Trabalho de conclusão de curso submetido ao Bacharelado em Ciência da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Antônio Carlos Mariani

Florianópolis (SC)

2014

À minha mãe, que mesmo não acreditando em nada, não desacreditou de mim...

AGRADECIMENTOS

Agradeço à minha família (magrese, dona norma, tio nilo, dona nadir, falecido seu belfort), que nunca desacreditou e nem teve pressa; à galera da ”esquina da olegário”, por me fornecer a ”envergadura moral”; à rapaziada do bosque, por provar que uma roda pode ser eterna; ao ”Barça”, professor Nereu e todos do Labcal, os primeiros a me adotar na UFSC; ao André Dyck, professor Mariani e todos do ”NPD/SeTIC” e ”Moodle UFSC”, pela segunda casa e segunda família na universidade; a todos que fizeram parte da ”Casa amerela” (original e beach) e da ”Turma do Moreira”, pelos porres homéricos e ”hospícios” no ”after”.

let us never regard a question as exhausted, and when we have used our last argument, let us begin again, if need be, with eloquence and irony - Proudhon

RESUMO

Os fóruns de discussão, também conhecidos como *bulletin boards*, são espaços na *Web* destinados ao compartilhamento de informações e discussão de tópicos e assuntos diversos. Nos últimos anos, o gigantesco grupo de usuários destes fóruns criou uma enorme base de conhecimento que ainda não é utilizada em seu todo seu potencial. Este trabalho apresenta o trabalho de integração da ferramenta de fórum implementada pelo ambiente virtual de aprendizagem Moodle com as ferramentas de aprendizado de máquina Apache Lucene e Apache Solr e com dois objetivos principais: detecção automática de spam e sugestão de discussões relacionadas.

Palavras-chave: Moodle. Aprendizado de máquina. Fórum. Solr. Lucene.

ABSTRACT

Discussion foruns, also known as bulleting boards, are web systems destinated to information share and discussion about any interesting topics. In last years, a big group of users have created an immense base of knowledge that is still not used at it's full potential. This work shows the integration of the forum system implemented by the e-learning environment Moodle with the machine learning tools Apache Lucene and Apache Solr two main objectives: automatic spam detection and reporting and related discussions suggestions.
Keywords: Moodle. Machine Learning. Forum. Solr. Lucene.

SUMÁRIO

1 INTRODUÇÃO

Os fóruns de discussão se tornaram uma ferramenta útil para a resolução de problemas (STEIN; MAIER, 1995), discussões de aprendizado (JENG et al., 2005), para gerenciar o conhecimento em organizações, como fóruns de cursos nas escolas, para edições de notícias (LICHTNER et al., 2009) (PENDERGAST, 2006) (UPDEGROVE; SMITH; BOLLENTIN, 1988) e também para suporte a softwares e demais produtos de várias empresas.

Uma das vantagens na utilização de fóruns online é que os usuários podem receber respostas adaptadas de outros usuários após formular problemas em suas próprias palavras, sem usar palavras-chave específicas para buscar (Steehouder, 2002).

Segundo (MUTHMANN et al., 2009), grande parte dos usuários dos fóruns não está satisfeita com as ferramentas de buscas atuais e acabam por incluir perguntas repetidas, o que dificulta ainda mais a busca convencional e aumenta consideravelmente o número de questões sem resposta. Além disso, segundo (XI; LIND; BRILL, 2004), a performance da busca em texto tradicional dos motores de busca não é adequada devido a atributos únicos dos fóruns de discussão e grupos de email.

Segundo (WU et al., 2012), os fóruns de discussão enfrentam as mesmas questões problemáticas que dizem respeito ao compartilhamento de conhecimento: ”quem é a pessoa (ou o canal) certa para perguntar?”; ”quando eu terei a resposta?”; ”como reduzir o tempo de solicitação?”.

Outro problema na busca por informações em fóruns é o fato das longas *threads* de discussão evoluírem ao longo do tempo para assuntos divergentes do original, o que impossibilita aos usuários adivinharem o conteúdo de uma discussão olhando apenas o título da *thread* (KIM; CANDAN; DöNDERLER, 2005).

Atual e historicamente este problema de encontrar discussões relacionadas é resolvido por membros mais antigos dos fóruns, que acompanham as discussões há algum tempo e têm maior facilidade em navegar no histórico de discussões para encontrar e indicar o link apropriado.

No caso de uma comunidade com mais de um fórum, os membros antigos ou membros ”mantenedores”também são responsáveis por garantir que os *posts* dos usuários sejam devidamente categorizados, movendo a discussão de um fórum para outro quando julgam apropriado ou até mesmo removendo ou bloqueando respostas a um tópico que não tenha relação com o espaço.

2 OBJETIVOS

No presente trabalho são apresentadas propostas de automatização (utilizando ferramentas que implementam técnicas de aprendizado de máquina (MITCHELL, 1997)) para duas tarefas importantes executadas pelos mantenedores de fóruns de discussão: detecção de spam e sugestão de discussões relacionadas.

Como ferramenta de fórum, foi escolhida a implementação presente no software Moodle, software livre que implementa um ambiente virtual de aprendizagem, com diversas ferramentas como a de fórum de discussão, que é utilizada pela comunidade de desenvolvedores e usuários ao redor do Moodle para interação desde a primeira publicação do software, contando com milhares de usuários e centenas de novos *posts* todos os dias.

Para complementar a ferramenta de fórum existente com as duas novas características, serão utilizados dois outros softwares livres: Apache Lucene, uma ferramenta de indexação, busca e diversos outros tratamentos de texto que fará a classificação dos *posts* em *spam*, os indesejados ou maliciosos, e *ham*, os legítimos ou desejáveis; e Apache Solr, uma ferramenta de indexação e busca em texto, que nos fornecerá a infraestrutura necessária para a sugestão de discussões relacionadas.

Desta forma, os objetivos deste trabalho são: mostrar que a aplicação de técnicas de *classificação* para detecção de spam é possível e pode ter uma boa influência no processo de manutenção de um fórum de discussão; facilitar a recuperação de informação por parte dos usuário e dessa forma diminuir a quantidade de conteúdo repetido e a necessidade de revisão por outros membros do fórum; mostrar que existem soluções maduras e de qualidade distribuídas como software livre; servir como inspiração para implementações semelhantes em outras plataformas de software de discussão; fornecer uma base para que novos algoritmos e técnicas sejam facilmente implementados nos fóruns de discussão do Moodle.

Este trabalho está organizado da seguinte maneira: O capítulo 2 apresenta os trabalhos relacionados. O capitulo 3 apresenta o Moodle, seus fóruns de discussão e as ferramentas Lucene e Solr. O capítulo 4 descreve o processo de integração entre o Moodle e as Ferramentas. O capítulo 5 mostra os resultados da integração com exemplos sobre dados reais. O capítulo 6 fala sobre trabalhos futuros.

3 TRABALHOS RELACIONADOS

Para tomar vantagem do conhecimento disponível nos fóruns de discussão, alguns pesquisadores começaram a construir redes especialistas e sistemas de buscas de perguntas e respostas para fóruns online (LI; LIAO; LAI, 2012).

Outros trabalhos utilizam algoritmos como PageRank dos links (XU; MA, 2006), mas isso traz diversos problemas para filtrar o conteúdo das páginas.

Outra abordagem, apresentada por (GOTTIPATI; LO; JIANG, 2011), é utilizar um motor que classifique os *posts* em perguntas, respostas, perguntas esclarecedoras, e respostas esclarecedoras e então fazer uma busca semântica baseada nessas tags atribuídas automaticamente. Este processo de classificação necessita que um volume suficiente de dados de treinamento seja classificado manualmente antes de iniciar o processo, o que dificulta a aplicação deste trabalho em ambientes de produção.

Além do conteúdo dos *posts*, outros trabalhos utilizam a estrutura das *threads* para melhorar a descoberta de informação. (DUAN; ZHAI, 2011) propõem dois esquemas de suavização para explorar as estruturas das *threads* nos fóruns auxiliado por técnicas de modelagem estatística de linguagem.

Ainda no contexto de recuperação da informação, (CONG et al., 2008) aborda o problema de encontrar pares de pergunta e resposta em fóruns de discussão.

No presente trabalho, esses problemas de separar o conteúdo interessante de outros itens da página não são aplicáveis pois é utilizado como fonte de dados diretamente o conteúdo dos *posts* dos usuários.

Um problema semelhante, tratado por (SINGH; P; RAGHU, 2012), é o de encontrar *threads* semelhantes em fóruns. Neste trabalho é proposta uma nova metodologia para estimar similaridadee entre pares de *threads* em fóruns de discussão que aproveita a informação estrutural das *threads* decompondo-as em componentes ponderados sobrepostos.

4 APRENDIZADO DE MÁQUINA

De acordo com (SAMUEL, 1959), o ”aprendizado de máquina”é ”o campo de estudos que dá aos computadores a habilidade de aprender sem serem explicitamente programados”. Tom Mitchell define de uma maneira um pouco mais formal: ”É dito que um programa de computador aprende sobre uma tarefa T, a partir de experiência E com alguma medidade de performance P se a sua performance em T, como medida por P, aumenta com experiência E.”(MITCHELL, 1997)

O aprendizado de máquina se propõe a resolver diversas classes de problemas, geralmente relacionados a reconhecimento de padrões e tomada inteligente de decisões baseados em dados de entrada. Isso envolve desde reconhecimento de texto manuscrito, detecção de spam, veículos automotivos não tripulados, sugestão de itens de interesse, classificação de itens etc.

4.1 CLASSIFICAÇÃO

Classificação é um processo de aprendizado supervisionado onde a máquina aprende a classificar futuras intâncias baseado num *corpus* previamente classificado. Neste trabalho utilizo a ferramenta Apache Lucene para classificar *posts* em duas categorias: spam, mensagens indesejadas; e ham, mensagens legítimas.

O processo de classificação depende de uma coleção previamente classificada. Como este trabalho trata de detecção de spam, utilizarei aqui uma lista aberta fornecida pelo projeto Spam Assasin (SPAMASSASSIN...,). Por convenção, 80% deste corpus é utilizado para treinar o classificador e os outros 20% para testar a qualidade da classificação.

4.2 SIMILARIDADE SEMÂNTICA E MEDIDAS SEMÂNTICAS

De acordo com (HARISPE et al., 2013), medidas semânticas são utilizadas atualmente para estimar a força da relação semântica entre elementos de vários tipos, por exemplo palavras, sentenças e documentods

Essas medidas semânticas generalizam as noções de similaridade semântica, relações semânticas e distância semântica, já muito discutidas na comunidade acadêmica.

O trabalho supracitado divide os modelos de medidas semânticas em três grupos: Medidas Distributivas, que estudam a repartição e a repetição

de unidades de linguagem em textos não estruturados; Medidas baseadas em conhecimento, que são aquelas que se apoiam em qualquer forma de representação de conhecimento, como, por exemplo, vocabulário estruturado ou ontologias ou grafos semânticos; e sistemas híbridos, que misturam as medidas baseadas em conhecimento com as medidas distributivas.

No presente trabalho, será utilizada uma técnica distributiva, chamada Vector Space Model - doravante VSM - (SALTON; WONG; YANG, 1975).

Em geral, a idéia por trás do VSM é quanto mais vezes um termo de uma consulta aparece em um documento relativo ao número de vezes que o termo aparece em todos os documentos da coleção, mais relevante aquele documento é para aquela consulta.

5 SOFTWARES UTILIZADOS

5.1 MOODLE

O Moodle é um sistema de gerenciamento de cursos, também conhecido como sistema gerenciador de aprendizado ou ainda ambiente virtual de ensino aprendizado. O Moodle se destaca por ser software livre (FOUNDATION,) e ser escrito em *PHP*, uma popular linguagem para sistemas web que permite a execução em várias plataformas.

A primeira versão do Moodle foi lançada em 20 de agosto de 2002 e hoje conta com mais de 70 mil sites registrados e verificados espalhados ao redor do mundo (MOODLE.ORG,). O Moodle reúne uma série de ferramentas típicas de ambientes virtuais de ensino-aprendizagem como fóruns, tarefas de upload de arquivos, wikis, chat, diversas formas de avaliação e acompanhamento de progresso do aluno num curso. Um fato interessante é que a ferramenta é utilizada para gerenciar a própria comunidade internacional de usuários e desenvolvedores, principalmente através de fóruns de discussão.

Nos primeiros dez anos de desenvolvimento, o Moodle alcançou mais de 200 países (MOODLE.ORG,), sendo que o Brasil é o terceiro colocado em número de sites registrados, com cerca de 4200 registros.

5.1.1 O plugin Fórum de discussão

Os fóruns da comunidade Moodle internacional cobrem diversas áreas relacionadas ao software desde a pedagogia e estratégias de uso das ferramentas até discussões de análises de novas funcionalidades ou problemas, tanto no código oficial quanto em diversas extensões desenvolvidas por terceiros. A maior parte do conteúdo está em inglês, mas existem também espaços reservados para as comunidades nacionais, com fóruns em mais de 20 idiomas.

A ferramenta de fórum do Moodle, apesar de ser uma peça importante, cumpre seu papel de forma apenas satisfatória. Uma evidência disso é a contribuição da Open University na forma do *plugin* ForumNG (FORUMNG,). Este plugin traz uma série de novas funcionalidades como edição com AJAX, salvar *posts* como rascunho e acompanhamento de *threads* individuais dentro de um fórum.

Dentre as diversas ferramentas disponíveis no Moodle, uma das mais utilizadas são os Fóruns. Esta ferramenta permite que os usuários tenham discussões e troca de idéias assíncronamente publicando comentários. O Mo-

odle oferece quatro tipos de fóruns, mas basicamente todos são compostos por uma ou mais *threads* (tópicos ou discussões), onde os usuários podem publicar *posts* (comentários). Cada comentário pode ter arquivos anexados e pode receber uma nota por usuários com permissão de avaliação.

A Figura 1 mostra as tabelas do banco de dados do Moodle e os relacionamentos da estrutura de fórum . Das 7 tabelas que compõem este contexto, esse trabalho utilizará apenas as 3 principais: *forum*, *forum_discussions* e *forum_posts*.

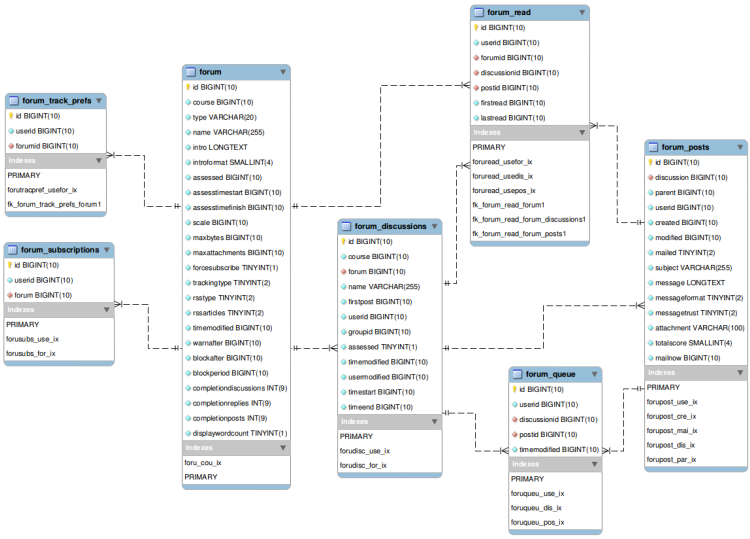

Figura 1: Tabelas do Moodle relativas ao forum

A tabela *forum* guarda as instâncias dos fóruns e suas configurações. Cada fórum existente em um curso no Moodle é representado por um registro nesta tabela. Por exemplo, no curso da comunidade brasileira no Moodle.org, existem os fóruns ”Avisos e novidades”, ”Resolução de problemas ténicos”, ”Tradutores”, dentre outros.

Um fórum reúne diversas discussões, também conhecidas como *threads*. No Moodle, cada discussão é representada por um registro na tabela *forum_discussions*. Por exemplo, no fórum ”Resolução de problemas técnicos”podemos encontrar discussões como ”Restrição por nota de atividade demora a ser liberada”, ”Instalação Moodle 2.5.1+ (problemas no login)”, ”Como habilitar apenas as categorias na página inicial dos administradores”, etc.

Por sua vez, cada discussão agrupa uma série de comentários de usuários ou *posts*. Estes comentários (bem como seus metadados), ficam armazenados na tabela *forum_posts*.

5.1.2 Busca nos fóruns de discussão

As funcionalidades de busca nos fóruns do Moodle, que podemos ver na Figura 2 são as tradicionais: filtros por período das datas da mensagem, fórum em que se encontra a mensagem, palavras que devem aparecer no texto, etc). Todos estes filtros demandam um conhecimento prévio do usuário para encontrar a informação desejada.

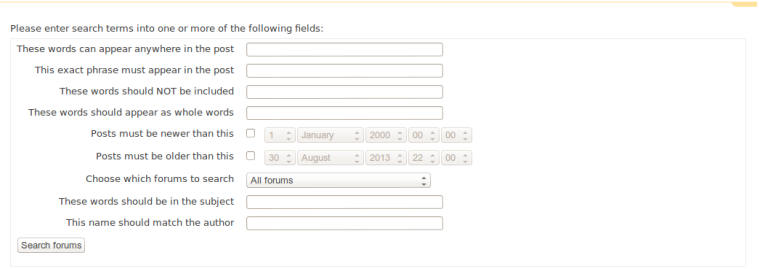

Figura 2: Formulário de busca nos fóruns do Moodle.

5.1.3 O plugin Global Search

Em maio de 2013, um desenvolvedor chamado Prateek Sachan iniciou o desenvolvimento da integração entre o Moodle e o Solr, ferramenta de indexação e busca em texto (WRITING. . . ,).

Esta integração tem como intuito fornecer uma interface unificada de busca no ambiente Moodle inteiro, ou seja, em todo o conteúdo incluído por professores e alunos, como fóruns, quizzes, atividades, arquivos anexados, etc (MOODLE. . . , a).

Este trabalho ainda está em andamento e não está disponível na distribuição oficial do Moodle, apesar de já ter sido revisado por alguns membros da comunidade e já ter sido considerado para inclusão nas próximas versões. O principal motivo da demora na integração à distribuição padrão é falta de uma melhor política de permissão de acesso aos documentos no momento da busca (MOODLE. . . , b).

5.2 APACHE LUCENE

O projeto Apache Lucene é uma iniciativa da Apache Fundation para fornecer softwares livres relacionados a busca em texto. Fazem parte deste projeto três softwares principais: o Lucene Core, uma ferramenta escrita em Java para indexação e busca textual, assim como verificação ortográfica e outras funcionalidades essenciais; o Solr, um servidor de aplicação para busca implementado em cima do Lucene Core, que fornece uma interface XML/HTTP e JSON/Python/Ruby APIs; por fim, pyLucene que é um *port* do projeto Core para linguagem Python.

Basicamente, dar suporte à *full-text search* utilizando a ferramenta Apache Lucene requer dois passos: criar um índice sobre os documentos ou objetos da base de dados; analisar a consulta do usuário e buscar no índice pré-construído por respostas para a consulta do usuário.

Atualmente existem quatro analisadores disponíveis para gerar os índices, resumidos na Tabela 1 (SMART, 2006).

Analisador	Descrição
StandardAnalyzer	Um analisador de propósito geral.
WhitespaceAnalyzer	Um analisador muito simples que apenas separa os tokens por espaços em branco.
StopAnalyzer	Remove palavras comuns do idioma inglês que geralmente não são úteis para indexação.
SnowballAnalyzer	Um analisador experimental que trabalha nas raízes das palavras (uma busca por ”chuva”também retornaria entradas para ”chovendo”, ”choveu”, etc).

Tabela 1: Analisadores disponíveis

Para fazer o trabalho de recomendação de documentos similares, o Lucene oferece a classe/funcionalidade *MoreLikeThis*. Esta classe é responsável por gerar uma consulta de busca ao índice do lucene que compara grandes corpos de texto automaticamente.

A classe *MoreLikeThis* permite que seja informado para o Lucene um documento já indexado e quais campos devem ser utilizados para calcular a similaridade (como, por exemplo, assunto e corpo dos*posts* de um fórum).

5.2.1 Busca e similaridade

O Lucene oferece uma API expansível para definir a similaridade entre documentos. A classe padrão, e a mais antiga oferecida pelo Lucene, é a classe *TFIDFSimilarity*, que implementa uma combinação do VSM para resgatar informação e o modelo Booleano (LANCASTER; FAYEN, 1973) para determinar o quão relevante um documento é para a consulta de um usuário (LUCENE. . . , b).

O Lucene adiciona algumas capacidades e refinamentos neste modelo para dar suporte a busca booleana e fuzzy, mas essencialmente permanece um sistema VSM em essência (LUCENE. . . , a).

A classe *MoreLikeThis*, utilizada para construir buscas para grandes corpos de texto, aplica ainda outras heurísticas ao processo de comparação de documentos para reduzir o conjunto de termos testados. De acordo com (DOUG, 2006), "sendo que o objetivo é maximizar a pontuação *tf*idf*, então o interesse está provavelmente em termos com um maior tf, logo, escolher um limiar para o tf mesmo baixo como 2 ou 3 reduzirá radicalmente o número de termos a se condirerar". Ainda outra heurística apresentada é que "termos com um alto *idf* tendem a ser longos, então é possível também colocar um limiar excluindo todos os termos da pesquisa que tem menos do que, por exemplo, seis ou sete caracteres".

5.2.2 Classificação de documentos

Para implementar o classificador de *spam*, será utilizada a API de classificação presente na ferramenta Lucene.

O processo de classificação do Lucene utiliza o próprio índice pré-indexado pela ferramenta de indexação de documentos, que é também base para os algoritmos de busca, para treinar a rede.

A vantagem de utilizar o Lucene para a classificação é a quantidade de ferramentas já presentes para fazer a tokenização, *stemming*, remoção de termos muito frequentes, criação dos vetores de termos, etc.

O Lucene oferece implementações de dois algoritmos de classificação clássicos: Naive Bayes (LEWIS, 1998) e Nearest Neighbor (ALTMAN, 1992). Neste trabalho será utilizado o primeiro algoritmo, por sua simplicidade de uso em relação ao segundo.

5.3 APACHE SOLR

Para extrair os dados da base do Moodle e indexá-los para uso pelas ferramentas de *clustering*, é utilizada a ferramenta Apache Solr (APACHE. . . ,), uma biblioteca de motor de busca e indexação de texto escrita em Java, também mantida pela Apache Foundation.

O Solr é escrito em Java, e apresenta seus serviços através de um container Servlet como o Jetty. O Solr utiliza a biblioteca Lucene para indexação e busca em textos e, através do Servlet, apresenta uma API HTTP REST (FIELDING; TAYLOR, 2002), podendo trabalhar com dados tanto em JSON (CROCKFORD,) quanto XML (BRAY et al.,).

Além desses serviços web, o Solr também possui um *frontend* para administração através da web (Figura 3), que é executado com o mesmo container dos serviços. Esta interface web permite coletar estatísticas sobre as diversas coleções de documentos armazenadas e indexadas pelo Solr e também interagir com essas coleções fazendo buscas.

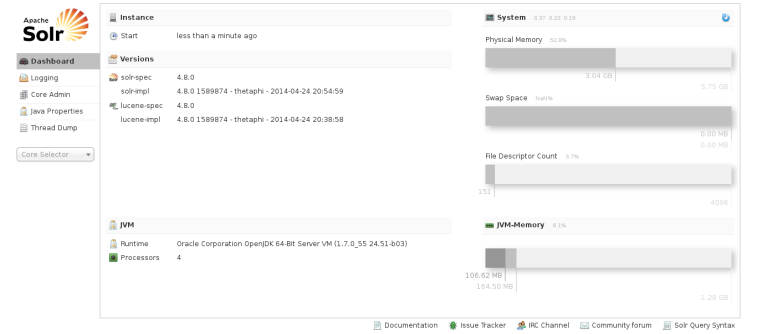

Figura 3: Página inicial da interface administrativa do Solr

As buscas no Solr são feitas com uma linguagem específica, definida pela *Solr Query Syntax* e permite especificar quais campos dos documentos buscar (título, conteúdo, ou outros campos quaisquer), em qual ordem os documentos devem ser apresentados, destacar os termos encontrados, influenciar no peso de cada termo da busca para similaridade, e utilizar alguns operadores lógicos.

Todos os processos relativos ao tratamento dos documentos recebi-

dos e indexados pelo Solr podem ser configurados sem a necessidade de programação em Java, bastando configurar um arquivo XML definindo quais classes específicas utilizar em cada parte do processo: do recebimento, tokenização, redução de termos aos ”radicais”(*stemming*), remoção de *stop words*, vetorização dos termos, cálculo de similaridade entre vetores, e a busca por termos.

Como o processo de indexação é dependente de idioma (principalmente a parte relativa à extração das raízes e remoção de *stopwords*), neste trabalho escrito um arquivo de configuração para utilizar as classes disponíveis para o tratamento do idioma português brasileiro e também foi utilizado uma lista de *stop words* disponibilizada pelo projeto Apache Lucene para o mesmo idioma.

6 INTEGRANDO MOODLE COM LUCENE E SOLR

A integração entre estes softwares foi feita em quatro partes: primeira, atualizar e adaptar o código do plugin Global Search; segunda, atualizar e adaptar o código do plugin Spam Deletion; terceira, desenvolvimento do código necessário para criar o serviço de detecção de *spam* com Lucene; quarta, desenvolvimento do código necessário para comunicação do Moodle com o Lucene e novas interações com o Solr.

6.1 ATUALIZAÇÃO DO PLUGIN GLOBAL SEARCH

Como o código do Global Search ainda não faz parte da distribuição oficial do Moodle e não foi alterado nos últimos 6 meses(GITHUB..., a), o primeiro passo da integração foi atualizá-lo para a versão 2.7 do Moodle. Isto foi feito com a ajuda da ferramenta git (GIT,), que permite reaplicar as diferenças dos arquivos (diffs, ou patches) de uma versão antiga e um versão mais nova, fazendo automaticamente os devidos deslocamentos. Após este "rebase", foram feitas pequenas mudanças para adequar às novas APIs do Moodle (GITHUB..., b).

Com o Global Search atualizado para funcionar na versão 2.7, o primeiro passo da integração é acertar o processo de envio de informaçõesdo Moodle para o Solr, que fará a indexação dos documentos.

Como o processo de indexação é dependente de idioma e o Global Search fornece apenas as configurações para inglês, foram criados os arquivos necessários para o idioma português: *stop words_br.txt* e *schema_br.xml*.

O primeiro arquivo define as *stop words* e faz parte da distribuição oficial do Lucene (LUCENE/ANALYSIS/BR/STOPWORDS...,).

No arquivo *schema_br.xml* são definidos os filtros de *stemming* e também a utilização das *stopwords* do idioma português brasileiro.

O envio de dados dos Moodle para o Solr é feito periodicamente pela ferramenta de *cron* do Moodle. Para tal, é criada uma conexão com o Solr e todos os documentos criados e alterados desde a última indexação são recuperados da base de dados do Moodle e enviados ao Solr através do método *addDocument*. Depois que todos os documentos são adicionados, o pedido de indexação é confirmado e iniciado através do método *commit*.

Nos experimentos realizados, o processo de adicionar os documentos, que no nosso caso são cada um dos cerca de 184 mil posts, durou 102 minutos, uma média de 1803 posts recebidos pelo Solr por minuto, ou 30 por segundo ou 0.03 segundos por post enviado.

Este agrupamento de *posts* se mostra interessante para executar o processo de atualização do índice em um ambiente de produção utilizando os recursos de *cron* do Moodle no qual diversos processamentos em lote sao executados. Dessa forma, a cada execução (ou n execuções) do *cron* do Moodle, poderia ser executada a rotina de atualização do índice do Solr com os últimos*posts* criados desde a última atualização, não precisando atualizar o índice no exato momento da criação do post.

Apesar do agrupamento de *posts* ser recomendado para ambientes de produção, a necessidade da execução do cron dificulta e atrasa o desenvolvimento e testes da ferramenta. Para contornar este problema e agregar o restante do código necessário para a detecção de spam e outros objetivos deste trabalho, foi criado um plugin chamado *local-solr*.

O primeiro teste foi feito com a base de fóruns da implantação para apoio aos cursos prenseciais de graduação e pós graduação da UFSC, conhecida como Moodle UFSC Presencial.

Esta base que agrega cursos desde 2009 e tem 37570 fóruns, que somam 127242 discussões que reunem 184004 posts diferentes. A distribuição de posts nas discussões está de acordo com a Tabela 2.

Número de posts na discussão	Número de discussões
1	114506
[02 , 05)	9270
[05 , 10)	1896
[10 , 20)	1022
[20 , 30)	328
[30 , 40)	124
[40 , 50)	65
[50 , 60)	43
[60 , 70)	15
[70 , 80)	10
[80 , 90)	4
[90 , 100)	2
[100, 212]	12

Tabela 2: Distribuição de posts nas discussões

6.2 ATUALIZAÇÃO DO PLUGIN SPAM DELETION

Atualmente, nos fóruns do Moodle.org é possível que um usuário reporte um *post* como Spam, para futura conferência dos responsáveis e possível

exclusão de conteúdo.

Este recurso de reportar conteúdo como spam também está disponível para outros comentários no site e para conteúdo em perfil de usuários através do plugin Spam Deletion.

O plugin foi desenvolvido por Dan Poltawski e David Mudrák em 2012 e a última versão disponível é de fevereiro de 2013, funcionando apenas na versão 2.4 do Moodle. Dessa forma, a primeira parte deste trabalho foi atualizar o código para funcionar com a versão 2.7 do Moodle.

Este plugin gera relatórios de spam para os administradores e tem também um algoritmo para diferenciar o peso do voto dos usuários. Este algoritmo leva em consideração o número de*posts* de um usuário. Quanto mais*posts* um usuário tem nos fóruns, maior será o peso do seu voto ao considerar um conteúdo como spam.

Este comportamento de reportar Spam para ser devidamente revisado por um humano é um comportamento desejado em nosso caso. Vamos utilizar a API deste plugin para reportar Spam baseado em nossa consulta ao Lucene no momento em que um usuário incluir um post.

6.3 CLASSIFICAÇÃO DE SPAM COM APACHE LUCENE

Basicamente, o processo de classificação consiste em fornecer para o Lucene um grande número de *posts* ou *documents* já classificados. Neste trabalho é utilizado o corpus fornecido pelo Spam Assassin(SPAMASSASSIN. . . ,).

A classe que representa da rede Bayesiana e possui os algoritmos de trainamento e atribuição de categoria (classificação) é a classe *SimpleNaiveBayesClassifier*. Para treinar a rede, é utilizado o método *train*, que recebe como parâmetro um *AtomicReader*, classe responsável por ler os índices gerados anteriormente pelo Lucene (que podem ter sido construídos a partir de documentos enviados através do Solr), o atributo do documento que será utilizado para comparação entre os documentos (no nosso caso, o corpo do post) o atributo do documento que contém a categoria pré-definida e, finalmente, o analizador de texto (responsável por aplicar a remoção de *stop words*, fazer a tokenização, etc).

Utilizando o Jetty (FOUNDATION; MEMBERS,), um *web server* e um *container* para *javax.servlet*, é possível criar um *webservice* para fornecer o serviço de classificação de spam.

Para compilar o projeto e resolver as dependências de projeto, foi utilizado a ferramenta Maven (TEAM, a). Com ela é possível especificar em um XML (BRAY et al.,) quais as dependências de bibliotecas que o pro-

jeto possui e elas são baixadas automaticamente ao executar o comando *mvn package*.

Para executar o servidor web de testes com nosso *webservice*, basta executar *mvn jetty:run* para que o serviço esteja disponível na via HTTP na porta 8080.

6.4 REPORTANDO SPAM AUTOMATICAMENTE NOS FÓRUNS

Para integrar o SpamClassifierServlet, criado na seção anterior, com o plugin Spam Deletion do Moodle, foi desenvolvido um novo plugin para o Moodle e por ser ele também o responsável por fazer a sugestão de tópicos relacionados como veremos no próximo capítulo, foi chamado *local-solr*.

O plugin criado utiliza a API de eventos e observadores fornecida pelo Moodle para enviar uma requisição HTTP POST para o SpamClassifierServlet no exato momento em que o usuário insere um um *post* no fórum do Moodle via a interface web padrão.

Para cumprir esta tarefa, foi escrita uma classe dentro plugin que observa o evento *mod_forum*
events
post_created e consulta o SpamClassifierServlet através da classe curl (TEAM, b) disponível no Moodle.

O conteúdo do *post* necessário para enviar ao SpamClassifierServlet está disponível através do método *get_record_snapshot* do objeto *mod_forum*
events
post_created_event, que é recebido como parâmetro no método associado ao evento de criação de *posts*.

A requisição para o SpamClassifierServlet envia os campos *subject* e *message*, concatenados, nesta ordem, informados pelo usuários no momento da criação do *post* no fórum.

A resposta do SpamClassifierServlet é então tratada e caso indique que o post é um spam, então é feita a chamada ao plugin *Spam Deletion* indicando o ocorrido. Caso contrário, nada é feito.

Para indicar um *post* como spam, basta criar um objeto a partir da classe *forum_post_spam* indicando o *id* do *post* e invocar o método *register_vote*, indicando o *id* do usuário que está indicando o post como spam. Nesta primeira versão, foi indicado o *id* do usuário administrador, mas em versões futuras pode ser alterado para utilizar um usuário específico relativo ao plugin.

Quando este voto é registrado, é enviado um email para todos os usuários do Moodle que possuem a *capability block/spam*

_deletion:viewspamreport com o conteúdo do post e as identificações de qual usuário criou o post e qual usuário indicou como *spam*.

6.5 SUGESTÃO DE TÓPICOS RELACIONADOS COM SOLR

A sugestão de tópicos relacionados foi implementada no plugin *local-solr* para, utilizando a mesma API de eventos citada na seção anterior, no momento em que um usuário visualiza uma discussão do fórum, apresentar uma lista de tópicos relacionados resultante de uma consulta ao Solr utilizando a API do Global Search.

A mesma classe de observadores de eventos utilizada pela ferramenta de anti-spam foi estendida com mais um métodos, dessa vez para observar o evento *mod_forum*
events
discussion_viewed, que será disparado quando um usuário visualizar uma disussão do fórum e nos dará acesso à página visualizada pelo usuário possibilitando incluir novos elementos através de Javascript.

O Moodle fornece uma API para que, no início ou no fim do carregamento da página possa ser invocada uma função javascript, que será executado no navegador do usuaŕio, e também permite que essa função receba parâmetros do PHP, que é executado no servidor, onde é executada a consulta por tópicos relacionados.

Essa interação com a página que será entregue para o usuário é feita através da varíavel global *$PAGE*, sendo que o método para incluir a chamada de função no javascript é *js_init_call*, que recebe como argumento a função a ser executada (que por padrão deve estar declarada no arquivo *module.js* no diretório raiz do plugin) e os parâmetros a serem repassados do PHP (lado do servidor) para o javascript (lado do cliente).

A função javascript que inclui o link para mostrar as discussões relacionadas e a lista com estas discussões foi nomeada *M.local_solr.show_related_discussions* conforme padrões de codificação do Moodle. O primeiro parâmetro desta função é sempre uma referência para a biblioteca YUI (YAHOO!; WORLD,) que ajuda na interação com o DOM (CONSORTIUM,), os parâmetros seguintes são aqueles vindos do php.

A função implementada recebe apenas um parâmetro adicional, que é uma lista de objetos representando as discussões relacionadas contendo como atributos o nome da discussão e a url, que serão utilizados para compor a lista de links.

Para obter a lista de discussões relacionadas, é feita uma consulta ao Solr, utilizando a API do *plugin Global Search*, informando como parâmetro

principal da busca o nome da discussão e filtrando apenas para buscar resultados nos fóruns.

Para realizar esta consulta, é necessário primeiro instanciar um cliente, através da inclusão do arquivo *$CFG-¿dirroot . '/search/' . $CFG-¿SEARCH_ENGINE . '/connection.php'*.

Depois, é executada a função *solr_execute_query*, que recebe como parâmetro o cliente recém criado e um objeto contendo os parâmetros para a busca. Neste caso, os parâmetros são: *queryfield*, contendo o nome da discussão que está sendo visualizada e *modulefilterqueryfield*, que restringe a busca apenas aos fóruns.

O resultado desta consulta retorna uma estrutura completa representando as discussões relacionadas em todos os fóruns do Moodle, contendo todos os campos presentes na tabela *forum_discussions* do Moodle.

A estrutura repassada para o javascript é filtrada para conter apenas o nome da discussão e o link para a visualização da discussão pela interface web do Moodle, que são os únicos parâmetros necessários para construir a lista de links.

7 RESULTADOS

A documentação da ferramentas Lucene e Solr, apesar de conterem poucos exemplos de integração com outras ferramentas, principalmente as não escritas em Java, cumprem muito bem o propósito de descrever/ensinar o uso de técnica de aprendizado de máquina para trabalhar com grandes volumes de texto.

Acerca da detecção de spam, foi alcançado o objetivo de detecção automática, com uma taxa de acerto aceitável. A ferramenta SpamClassifierServlet, que utiliza os índeces do Solr e a bibliteca do Lucene ficou totalmente desacoplada do Moodle, o que permite reutilizar a solução para outras ferramentas, desenvolvidas em qualquer ligugagem, com qualquer propósito, que tenham necessidades de classifição de conteúdo como spam.

A parte sobre recomendação de discussões relacionadas também teve seus objetivos alcançados, mostrando-se uma solução prática e adotável, sem aumentar a carga sobre a base de dados do Moodle nem sobre o servidor web, pois toda a parte de indexação e busca é efetuada pelo Solr. A solução de apresentação de discussões para o usuário mostrou-se também não intrusiva e eficaz ao alterar minimamente a interface dos fóruns.

8 CONCLUSÃO

O presente trabalho mostrou que existem ferramentas maduras e eficientes para aplicar aprendizado de máquina em problemas de recuperação de informação e classificação de documentos.

Estas ferramentas fornecem os algoritmos mais conhecidos para tratamento de texto mas permitem também a extensão e implementação de novos algoritmos, utilizando-se de APIs padronizadas.

Com isso, a detecção de spam, característica comum nos serviços de email, tende a se tornar algo mais presente nas ferramentas de fórum, assim como a sugestão de conteúdo relacionado deixa de ser um diferencial apenas das ferramentas proprietárias e dos portais de vendas e entretenimento.

9 TRABALHOS FUTUROS

O presente trabalho desenvolveu a infraestrutura básica para a classificação de spam e recomendação de discussões relacionadas, podendo esta ainda ser largamente melhorada.

Na parte da detecção de spam, o processo de treinamento da rede pode ser executado de forma periódica, ao invés de a cada requisição, o que permite um menor tempo de resposta do classificador ainda menor com quase nenhuma perda acurácia, caso a massa de dados de treinamento não mude frequentemente.

O processo de classificação de spam poderia ser integrado ao Solr como um *QueryHandler* (ao invés de ser apresentado como um Servlet) e a parte de treinamento pode também ser integrada ao Solr como um *UpdateProcessor*, e utilizar o mesmo índice (ou apenas o conteúdo) das discussões do Moodle que foram previamente classificadas pelos usuários para treinar o classificador.

Os fóruns do Moodle são, talvez, a ferramenta colaborativa mais utilizada dentro da plataforma, mas outras ferramentas como a Wiki ou o Glossário também poderiam se beneficiar de filtros de spam, bem como da sugestão de conteúdo relacionado, uma vez que o tipo de técnica de recomendação utilizado extrapola os termos vistos nas páginas e utilizados nas buscas.

Além de integração com outras ferramentas, dentro ou fora do Moodle, trabalhos futuros podem medir o impacto da sugestão de discussões relacionadas através de estudo do perfil dos usuários (analisando as buscas relacionadas e/ou as páginas visitadas) ou do histórico de criação de novas discussões e taxa de repetiação.

O presente trabalho utilizou apenas técnicas de similaridade semântica baseadas em distribuição estatísticas dos termos e, mesmo neste campo, estão longe de se esgotar as possibilidades de algoritmos e técnicas possíveis de aplicação em fóruns de discussão ou outros meios, de forma que a comparação dos resultados deste trabalho com a aplicação de outras técnicas distributivas ou aquelas baseadas em conhecimento seria de grande contribuição para este campo de pesquisa.

LISTA DE FIGURAS

REFERÊNCIAS BIBLIOGRÁFICAS

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, v. 46, n. 3, p. 175–185, 1992. <http://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879>.

APACHE Lucene. <http://lucena.apache.org/solr>. Acessado em 05 junho 2014.

BRAY, T. et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. <http://www.w3.org/TR/REC-xml>. Acessado em 05 junho 2014.

CONG, G. et al. Finding question-answer pairs from online forums. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2008. (SIGIR '08), p. 467–474. ISBN 978-1-60558-164-4. <http://doi.acm.org/10.1145/1390334.1390415>.

CONSORTIUM, W. W. W. *Document Object Model*. <http://www.w3.org/DOM>. Acessado em 05 junho 2014.

CROCKFORD, D. *The JavaScript Object Notation (JSON) Data Interchange Format*. <http://tools.ietf.org/html/rfc7159>. Acessado em 05 junho 2014.

DOUG. *Lucene -More Like This Feature*. 2006. <http://lucene.apache.org/core/4_8_0/queries/org/apache/lucene/queries/mlt/MoreLikeT Acessado em 05 junho 2014.

DUAN, H.; ZHAI, C. Exploiting thread structures to improve smoothing of language models for forum post retrieval. In: *Proceedings of the 33rd European conference on Advances in information retrieval*. Berlin, Heidelberg: Springer-Verlag, 2011. (ECIR'11), p. 350–361. ISBN 978-3-642-20160-8. <http://dl.acm.org/citation.cfm?id=1996889.1996935>.

FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, ACM, New York, NY, USA, v. 2, n. 2, p. 115–150, maio 2002. ISSN 1533-5399. <http://doi.acm.org/10.1145/514183.514185>.

FORUMNG. <http://docs.moodle.org/25/en/ForumNG>. Acessado em 05 junho 2014.

FOUNDATION, I. F. S. *The Free Software Definition*. <http://www.gnu.org/philosophy/free-sw.html>. Acessado em 05 junho 2014.

FOUNDATION, T. E.; MEMBERS comunnity. *Jetty, Servlet Engine and HTTP Server*. <http://maven.apache.org/>. Acessado em 05 junho 2014.

GIT. <http://www.git-scm.com>. Acessado em 05 junho 2014.

GITHUB - Global Search. <https://github.com/prateeksachan/moodle/tree/gs2>. Acessado em 05 junho 2014.

GITHUB - Global Search. <https://github.com/danielneis/moodle/compare/MDL-31989>. Acessado em 05 junho 2014.

GOTTIPATI, S.; LO, D.; JIANG, J. Finding relevant answers in software forums. In: *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*. [S.l.: s.n.], 2011. p. 323–332. ISSN 1938-4300.

HARISPE, S. et al. Semantic measures for the comparison of units of language, concepts or entities from text and knowledge base analysis. *CoRR*, abs/1310.1285, 2013.

JENG, Y.-L. et al. Ants: Agent-based navigational training system. In: LAU, R. et al. (Ed.). *Advances in Web-Based Learning - ICWL 2005*. Springer Berlin Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3583). p. 320–325. ISBN 978-3-540-27895-5. <http://dx.doi.org/10.1007/11528043_32>.

KIM, J. W.; CANDAN, K. S.; DöNDERLER, M. E. Topic segmentation of message hierarchies for indexing and navigation support. In: *Proceedings of the 14th international conference on World Wide Web*. New York, NY, USA: ACM, 2005. (WWW '05), p. 322–331. ISBN 1-59593-046-9. <http://doi.acm.org/10.1145/1060745.1060795>.

LANCASTER, F. W.; FAYEN, E. G. Book. *Information retrieval: on-line [by] F. W. Lancaster and E. G. Fayen*. [S.l.]: Melville Pub. Co Los Angeles, 1973. xiv, 597 p. p. ISBN 0471512354.

LEWIS, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. In: *Proceedings of the 10th European Conference on Machine Learning*. London, UK, UK: Springer-Verlag, 1998. (ECML '98), p. 4–15. ISBN 3-540-64417-2. <http://dl.acm.org/citation.cfm?id=645326.649711>.

LI, Y.-M.; LIAO, T.-F.; LAI, C.-Y. A social recommender mechanism for improving knowledge sharing in online forums. *Inf. Process. Manage.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 48, n. 5, p. 978–994, set. 2012. ISSN 0306-4573. <http://dx.doi.org/10.1016/j.ipm.2011.10.004>.

LICHTNER, V. et al. An online forum as a user diary for remote workplace evaluation of a work-integrated learning system. In: *CHI '09 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2009. (CHI EA '09), p. 2955–2970. ISBN 978-1-60558-247-4. <http://doi.acm.org/10.1145/1520340.1520424>.

LUCENE - Scoring. <http://lucene.apache.org/core/2_9_4/scoring.html>. Acessado em 05 junho 2014.

LUCENE - Similariry. <https://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/Simila Acessado em 05 junho 2014.

LUCENE/ANALYSIS/BR/STOPWORDS.TXT. <http://svn.apache.org/repos/asf/lucene/dev/trunk/lucene/analysis/common/src/resource Acessado em 05 junho 2014.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw Hill, 1997. 2 p.

MOODLE Docs - Global Search. <http://docs.moodle.org/dev/Global_search>. Acessado em 05 junho 2014.

MOODLE Tracker - Global Search. <https://tracker.moodle.org/browse/MDL-31989>. Acessado em 05 junho 2014.

MOODLE.ORG. *Moodle Statistics*. <http://moodle.org/stats>. Acessado em 05 junho 2014.

MUTHMANN, K. et al. Near-duplicate detection for web-forums. In: *Proceedings of the 2009 International Database Engineering & Applications Symposium*. New York, NY, USA: ACM, 2009. (IDEAS '09), p. 142–151. ISBN 978-1-60558-402-7. <http://doi.acm.org/10.1145/1620432.1620447>.

PENDERGAST, M. An analysis tool for the assessment of student participation and implementation dynamics in online discussion forums. *SIGITE Newsl.*, ACM, New York, NY, USA, v. 3, n. 2, p. 10–17, jun. 2006. ISSN 2166-1685. <http://doi.acm.org/10.1145/1142152.1142153>.

SALTON, G.; WONG, A.; YANG, C. S. A vector space model for automatic indexing. *Commun. ACM*, ACM, New York, NY, USA, v. 18, n. 11, p. 613–620, nov. 1975. ISSN 0001-0782. <http://doi.acm.org/10.1145/361219.361220>.

SAMUEL, A. *Machine Learning*. [S.l.: s.n.], 1959.

SINGH, A.; P, D.; RAGHU, D. Retrieving similar discussion forum threads: a structure based approach. In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2012. (SIGIR '12), p. 135–144. ISBN 978-1-4503-1472-5. <http://doi.acm.org/10.1145/2348283.2348305>.

SMART, J. F. *A Short Introduction to Lucene*. april 2006. <http://oak.cs.ucla.edu/cs144/projects/lucene/index.html>. Acessado em 05 junho 2014.

SPAMASSASSIN public mail corpus. <http://spamassassin.apache.org/publiccorpus/readme.html>. Acessado em 05 junho 2014.

Steehouder, M. F. Beyond technical documentation: users helping each other. In: *IEEE International Professional Communication Conference, IPCC 2002*. IEEE, 2002. p. 489–499. <http://doc.utwente.nl/55875/>.

STEIN, A.; MAIER, E. Structuring collaborative information-seeking dialogues. *Knowledge-Based Systems*, v. 8, p. 82–93, 1995.

TEAM, T. A. M. *Apache Maven Project*. <http://maven.apache.org/>. Acessado em 05 junho 2014.

TEAM, T. P. D. *PHP Curl Library*. <http://www.php.net//manual/en/book.curl.php>. Acessado em 05 junho 2014.

UPDEGROVE, D. A.; SMITH, S. B.; BOLLENTIN, W. R. Ccnews: an online forum for newsletter editors. In: *Proceedings of the 16th annual ACM SIGUCCS Conference on User Services*. New York, NY, USA: ACM, 1988. (SIGUCCS '88), p. 351–358. ISBN 0-89791-286-1. <http://doi.acm.org/10.1145/62548.62656>.

WRITING Moodle's Global Search. <https://moodle.org/mod/forum/discuss.php?d=227805>. Acessado em 05 junho 2014.

WU, H. C. et al. A split-list approach for relevance feedback in information retrieval. *Inf. Process. Manage.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 48, n. 5, p. 969–977, set. 2012. ISSN 0306-4573. <http://dx.doi.org/10.1016/j.ipm.2012.03.007>.

XI, W.; LIND, J.; BRILL, E. Learning effective ranking functions for newsgroup search. In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2004. (SIGIR '04), p. 394–401. ISBN 1-58113-881-4. <http://doi.acm.org/10.1145/1008992.1009060>.

XU, G.; MA, W.-Y. Building implicit links from content for forum search. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2006. (SIGIR '06), p. 300–307. ISBN 1-59593-369-7. <http://doi.acm.org/10.1145/1148170.1148224>.

YAHOO!, F. engineers at; WORLD contributors from around the. *YUI Library*. <http://yuilibrary.com/>. Acessado em 05 junho 2014.

APÊNDICE A – Código-fonte: SpamClassifierServlet

Arquivo: spamclassifierservlet/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchem
         a-instance"
         xsi:schemaLocation="http://maven.apache.
         org/POM/4.0.0
         http://maven.apache.org/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>org.apache.lucene</groupId>
    <artifactId>SpamClassifier</artifactId>
    <version>0.0</version>
    <packaging>jar</packaging>
    <name>Spam Classifier</name>

    <properties>
        <jettyVersion>7.2.0.v20101020</jettyVersion>
    </properties>
    <dependencies>
        <dependency>
          <groupId>org.eclipse.jetty</groupId>
          <artifactId>jetty-server</artifactId>
          <version>${jettyVersion}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.lucene</groupId>
            <artifactId>lucene-core</artifactId>
            <version>4.8.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.lucene</groupId>
            <artifactId>
            lucene-analyzers-common
            </artifactId>
            <version>4.8.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.lucene</groupId>
            <artifactId>
            lucene-classification
```

```
            </artifactId>
            <version>4.8.1</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <artifactId>
                maven-compiler-plugin
                </artifactId>
                <configuration>
                        <source>1.6</source>
                        <target>1.6</target>
                        <archive>
                            <manifest>
                                <addClasspath>
                                true
                                </addClasspath>
                                <classpathPrefix>
                                lib/
                                </classpathPrefix>
                                <mainClass>
                                org.apache.lucene.
                                SpamClassifier
                                </mainClass>
                            </manifest>
                        </archive>
                </configuration>
            </plugin>
            <plugin>
              <!-- This plugin is needed for the
              servlet example -->
              <groupId>org.mortbay.jetty</groupId>
              <artifactId>
              jetty-maven-plugin
              </artifactId>
              <version>${jettyVersion}</version>
            </plugin>
            <plugin>
              <groupId>org.codehaus.mojo</groupId>
              <artifactId>
```

```
                exec-maven-plugin
                </artifactId>
                <version>1.1</version>
                <executions>
                  <execution>
                      <goals>
                          <goal>java</goal>
                      </goals>
                  </execution>
                </executions>
                <configuration>
                  <mainClass>
                  org.apache.lucene.SpamClassifier
                  </mainClass>
                </configuration>
              </plugin>
            </plugins>
        </build>

</project>
```

Arquivo:
spamclassifierservlet/src/main/\
webapp/WEB-INF/web.xml

```
<?xml version="1.0" encoding="utf8"?>
<web-app
          xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-
          instance"
          xsi:schemaLocation="http://java.sun.com/xml/
          ns/javaee http://java.sun.com/xml/ns/javaee/
          web-app_2_5.xsd"
          version="2.5">
      <servlet>
          <servlet-name>SpamClassifier</servlet-name>
          <servlet-class>
          org.apache.lucene.SpamClassifier
          </servlet-class>
      </servlet>
      <servlet-mapping>
```

```
        <servlet-name>SpamClassifier</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

Arquivo:
spamclassifierservlet/src/main/\
java/org/apache/lucene/SpamClassifier.java

```
package org.apache.lucene;

import java.io.BufferedReader;
import java.lang.StringBuffer;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.util.Map;
import java.util.HashMap;

import java.io.File;
import java.io.IOException;

import org.apache.lucene.document.Document;

import org.apache.lucene.index.IndexReader;
import org.apache.lucene.index.AtomicReader;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.
        SlowCompositeReaderWrapper;

import org.apache.lucene.analysis.standard.
        ClassicAnalyzer;

import org.apache.lucene.classification.
        SimpleNaiveBayesClassifier;
import org.apache.lucene.classification.
        ClassificationResult;

import org.apache.lucene.store.*;
```

```
import org.apache.lucene.util.Version;
import org.apache.lucene.util.BytesRef;

public class SpamClassifier extends HttpServlet {

    protected void doPost(HttpServletRequest
                          request,
                          HttpServletResponse
                          resp)
            throws ServletException, IOException {

        StringBuffer postText = new StringBuffer();

        String line = null;

        try {
            BufferedReader reader = request.
                                        getReader();
            while ((line = reader.readLine())
                    != null)
                postText.append(line);
        } catch (Exception e) {
            System.out.println(e.toString());
            /*report an error*/
        }

        final long startTime = System.
                                currentTimeMillis();

        SimpleNaiveBayesClassifier classifier = new
                        SimpleNaiveBayesClassifier();

        String indexDir1 = "/solr-4.8.0/example/\
                            solr/collection1/\
                            data/index";

        DirectoryReader reader = DirectoryReader.
        open(FSDirectory.open(new File(indexDir1)));
        AtomicReader ar =
            SlowCompositeReaderWrapper.wrap(reader);
```

```
        classifier.train(ar, "text", "cat",
        new ClassicAnalyzer(Version.LUCENE_CURRENT));

        ClassificationResult<BytesRef> result =
        classifier.assignClass(postText.toString());
        String classified =
            result.getAssignedClass().utf8ToString();

        reader.close();

        resp.getWriter().print(classified);
    }
}
```

APÊNDICE B – Código-fonte: moodle-local-solr

Arquivo: moodle-local_solr/version.php

```
<?php

/**
 * Forum improvements using Apache Solr version
 * information
 *
 * @package    local_solr
 * @copyright 2014 Daniel Neis Araujo
 */

defined('MOODLE_INTERNAL') || die();

$plugin->version    = 2014061101;
$plugin->requires   = 2014050800;
$plugin->component = 'local_solr';
$plugin->cron       = 0;
$plugin->depencies = array('block_spam_deletion' =>
                                2014060200);
```

Arquivo: moodle-local_solr/settings.php

```
<?php

/**
 * Improvements to forum using Apache Solr
 * settings and presets.
 *
 * @package     local_solr
 * @copyright  2014 Daniel Neis
 */

defined('MOODLE_INTERNAL') || die();

if ($ADMIN->fulltree) {
    $settings =
        new admin_settingpage('local_solr_settings',
                get_string('settings', 'local_solr'),
                                'local/solr:config');

    $settings->add(new
        admin_setting_heading(
```

```
            'local_solr_settings_header', '',
            get_string('pluginname_desc',
                'local_solr')));

    $settings->add(new
        admin_setting_configtext(
            'local_solr/spamclassifierhost',
            get_string('spamclassifierhost',
                'local_solr'),
            get_string('spamclassifierhost_desc',
                'local_solr'),
            'localhost'));
    $settings->add(new
        admin_setting_configtext(
            'local_solr/spamclassifierport',
            get_string('spamclassifierport',
                'local_solr'),
            get_string('spamclassifierport_desc',
                'local_solr'),
            '8080'));

    $ADMIN->add('localplugins', $settings);
}
```

Arquivo: moodle-local_solr/module.js

```
M.local_solr = {};

// This function adds the 'show related discussions'
// link and list on every discussion.
M.local_solr.show_related_discussions =
  function(Y, related_discussions) {
    var mainsection =
        Y.one('#page-mod-forum-discuss #region-main');

    mainsection.append('<h3>' +
                M.str.local_solr.related_discussions +
                '</a></h3>');

    var related_list = '<ul id="relateddiscussions">';
    for (var i in related_discussions) {
        discussion = related_discussions[i];
        related_list += '<li><a href="' +
```

```
                    M.cfg.wwwroot +
                    discussion.link +
                    '">'+discussion.name +'</a></li>';
    }
    related_list += '</ul>';
    mainsection.append(related_list);
}
```

Arquivo: moodle-local_solr/classes/forum_observers.php

```
<?php

/**
 * Forum observers.
 *
 * @package     local_solr
 * @copyright   2014 Daniel Neis Araujo
 */

namespace local_solr;
defined('MOODLE_INTERNAL') || die();

class forum_observers {

    /**
     * Add a new post to Solr
     *
     * @param \mod_forum\event\post_created $event
     *                                      The event.
     * @return void
     */
    protected static function addToSolr(
            \mod_forum\event\post_created $event) {
        global $CFG;
        require_once($CFG->dirroot . '/search/' .
             $CFG->SEARCH_ENGINE . '/connection.php');
        require_once($CFG->dirroot .
                                '/search/lib.php');
        $search_engine_installed =
                    $CFG->SEARCH_ENGINE . '_installed';
        $search_engine_check_server =
                $CFG->SEARCH_ENGINE . '_check_server';
```

```
        if ($search_engine_installed() and
            $search_engine_check_server($client)) {
            try {
                $docs =
                    forum_search_get_documents(
                                    $event->objectid);
                foreach ($docs as $document) {
                    if ($document->
                        getField('type')->
                        values[0] == SEARCH_TYPE_HTML) {

                        $client->
                            add_document($document);
                    }
                }
                $client->commit();
            } catch (Exception $e) {
            }
        }
    }

    /**
     * A new post was created
     *
     * @param \mod_forum\event\post_created $event
     *                                      The event.
     * @return void
     */
    public static function post_created(
                \mod_forum\event\post_created $event) {

        self::addToSolr($event);

        self::classifyAndReportSpam($event);
    }

    /**
     * Classify a post as spam (and report it)
     * or ham
     *
     * @param \mod_forum\event\post_created $event
```

```
     *                                          The event.
     * @return void
     */
    protected static function classifyAndReportSpam(
                \mod_forum\event\post_created $event) {

        $snapshot =
            $event->get_record_snapshot('forum_posts',
                                        $event->objectid);
        $text = $snapshot->subject .' '.
                $snapshot->message;

        $config = get_config('local_solr');
        $curl = new \curl();
        $result =
            $curl->post($config->spamclassifierhost.
                        ':'.
                        $config->spamclassifierport,
                        array('text' => $text));
        if ($result_decoded = json_decode($result))
{
            if ($result_decoded['cat'] == 'spam') {
                require_once($CFG->dirroot .
                     '/blocks/spam_deletion/lib.php');
                $postspam =
                    new \forum_post_spam(
                                $event->objectid);
                $postspam->register_vote(2);
            }
        }
    }

    /**
     * A discussion was viewed
     *
     * @param \mod_forum\event\discussion_viewed $event
     *                                          The event.
     * @return void
     */
    public static function discussion_viewed(
          \mod_forum\event\discussion_viewed $event) {
```

```
global $CFG, $DB;

require_once($CFG->dirroot.'/search/' .
     $CFG->SEARCH_ENGINE.'/connection.php');
require_once($CFG->dirroot.'/search/lib.php');

$search_engine_installed =
          $CFG->SEARCH_ENGINE . '_installed';
$search_engine_check_server =
        $CFG->SEARCH_ENGINE . '_check_server';

if ($search_engine_installed() and
    $search_engine_check_server($client)) {

    $params[] = $event->objectid;
    $snapshot =
        $event->get_record_snapshot(
                        'forum_discussions',
                        $event->objectid);
    $firstpost =
        clean_param(
            $DB->get_field('forum_posts',
                           'message',
                           array(
                             'id' =>
                               $snapshot->
                               firstpost)
                           ),
            PARAM_TEXT);

    try {
        $query = new \SolrQuery();
        solr_add_fields($query);
        $query->setMlt(true);
        $query->setMltCount(5);
        $query->addMltField('content');
        $query->setQuery('"'.$firstpost.'"');
        $query->setStart(0);
        $query->setRows(10);
        $query->setMltMinDocFrequency(1);
        $query->setMltMinTermFrequency(1);
```

```
                $query->setMltMinWordLength(4);
                $query->setOmitHeader(5);
                $query_response =
                            $client->query($query);
                $response =
                       $query_response->getResponse();
                if ($mlt = (array)
                     $response->moreLikeThis) {
                    $mlt = array_pop($mlt);

                    if ($mlt->numFound > 0) {
                        $cleanresults = array();
                        foreach ($mlt->docs as $r){
                            $link =
                              substr($r->contextlink,
                                     0,
                                     strpos(
                                       $r->contextlink,
                                       '#'));
                            $discussion =
                              substr($link,
                                     strpos($link,
                                            '=') + 1);
                            $cleanresults[$discussion]
                             = array('name' =>
                                        $r->title,
                                     'link' => $link);
                        }
                        global $PAGE;
                        $PAGE->
                         requires->
                          strings_for_js(
                            array(
                              'related_discussions'),
                              'local_solr');
                        $PAGE->
                        requires->
                        js_init_call(
                        'M.local_solr.
____________________show_related_discussions',
                        array($cleanresults), true);
```

```
                    }
                }
            } catch (Exception $e) {
            }
        }
    }
}
```

Arquivo: moodle-local_solr/db/access.php

```
<?php
/**
 * Capabilities for Forum improvements
 * using Apache Solr
 *
 * @package     local_solr
 * @copyright   2014 Daniel Neis
 */

defined('MOODLE_INTERNAL') || die();

$capabilities = array(

    'local/solr:config' => array(
        'captype' => 'write',
        'contextlevel' => CONTEXT_COURSE,
        'archetypes' => array(
            'manager' => CAP_ALLOW,
        )
    ),

);
```

Arquivo: moodle-local_solr/db/events.php

```
<?php

$observers = array(

    array(
        'eventname'   =>
                    '\mod_forum\event\post_created',
        'callback'    =>
          '\local_solr\forum_observers::post_created',
    ),
```

```
    array(
        'eventname'   =>
                '\mod_forum\event\discussion_viewed',
        'callback'    =>
            '\local_solr\forum_observers::
␣␣␣␣␣␣␣␣␣␣␣␣␣discussion_viewed',
    ),
);
```

Arquivo: moodle-local_solr/lang/en/local_solr.php

```
<?php

$string['pluginname'] = 'Forum␣improvements
using␣Solr';
$string['pluginname_desc'] = 'Add␣automatic␣spam
reporting␣and␣related␣discussions␣suggestions␣to
forums␣using␣Apache␣Solr.';
$string['spamclassifierhost'] =
                        'SpamClassifierServlet␣Host';
$string['spamclassifierhost_desc'] =
'This␣is␣the␣full␣URL␣to␣the␣webservice␣provided
by␣SpamClassifierServlet.';
$string['spamclassifierport'] =
                        'SpamClassifierServlet␣Port';
$string['spamclassifierport_desc'] =
'The␣port␣to␣access␣the␣webservice␣provided
by␣SpamClassifierServlet.␣Default␣is␣8080.';
$string['settings'] =
                    'SpamClassifierServlet␣Settings.';
$string['related_discussions'] =
                                'Related␣discussions';
```