PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Susana Rosich Soares Velloso

SQLLOMining: Obtenção de Objetos de Aprendizagem utilizando técnicas de Aprendizado de Máquina

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Rubens Nascimento Melo

Rio de Janeiro, Julho de 2007

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Susana Rosich Soares Velloso

SQLLOMining: Obtenção de Objetos de Aprendizagem utilizando técnicas de Aprendizado de Máquina

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Rubens Nascimento Melo
Orientador
PUC-Rio

Ruy Luiz Milidiú
Departamento de Informática - PUC-Rio

Sérgio Lifschitz
Departamento de Informática - PUC-Rio

José Eugênio Leal
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 20 de julho de 2007

Susana Rosich Soares Velloso

Graduou-se em Engenharia Mecânica na PUC-Rio em 1994. Atuou como consultora em Banco de Dados para diversas empresas. Possui interesse acadêmico e profissional nas áreas de Inteligência Artificial e Banco de Dados.

Ficha Catalográfica

Velloso, Susana Rosich Soares

SQLLOMining: Obtenção de objetos de aprendizagem utilizando técnicas de aprendizado de máquina / Susana Rosich Soares Velloso; orientador: Rubens Nascimento Melo. – 2007.

118 f.; 30 cm

Dissertação (Mestrado em Informática)–Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

Inclui bibliografia

1. Informática – Teses. 2. Educação baseada na Web. 3. Objetos de aprendizagem. 4. Ontologia. 5. Banco de dados para e-Learning. 6. Aprendizado de máquina. 7. Classificação de textos. 8. Naive-Bayes. I. Melo, Rubens Nascimento. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Este trabalho é dedicado ao meu marido
pelo amor e companheirismo.

Agradecimentos

À PUC-Rio, ao departamento de informática e ao CAPES pela oportunidade.

Ao meu orientador, Rubens Nascimento Melo, pela motivação, paciência e ajuda.

Aos meus companheiros e companheiras de TecBD, pela motivação e ajuda.

A todos os professores, funcionários do Departamento de Informática pelo apoio dado quando precisei.

Ao professor Ruy Luiz Milidiú pelo apoio e pelo curso "Text Mining", grande inspiração para a minha dissertação.

Ao professor Eduardo Sany Laber pelo curso de PAA e pela carta de recomendação para o meu ingresso no departamento.

Às minhas amigas pelos momentos de pura diversão e pelo incentivo nos momentos difíceis.

Aos meus familiares pela educação e pelo apoio.

Aos professores que participaram da banca examinadora.

Resumo

Velloso, Susana Rosich Soares. **SQLLOMining: Obtenção de Objetos de Aprendizagem utilizando técnicas de Aprendizado de Máquina.** Rio de Janeiro, 2007. 118p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Objetos de Aprendizagem ou Learning Objects (LOs) são porções de material didático tais como textos que podem ser reutilizados na composição de outros objetos maiores (aulas ou cursos). Um dos problemas da reutilização de LOs é descobri-los em seus contextos ou documentos texto originais tais como livros, e artigos. Visando a obtenção de LOs, este trabalho apresenta um processo que parte da extração, tratamento e carga de uma base de dados textual e em seguida, baseando-se em técnicas de aprendizado de máquina, uma combinação de EM (Expectation-Maximization) e um classificador Bayesiano, classifica-se os textos extraídos. Tal processo foi implementado em um sistema chamado "SQLLOMining", que usa SQL como linguagem de programação e técnicas de mineração de texto na busca de LOs.

Palavras-chave

Educação Baseada na Web; Objetos de Aprendizagem; Ontologia; Banco de Dados para e-Learning; Aprendizado de Máquina; Classificação de Textos; Naive-Bayes

Abstract

Velloso, Susana Rosich Soares. **SQLLOMining: Finding Learning Objects using machine learning methods.** Rio de Janeiro, 2007. 118p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Learning Objects (LOs) are pieces of instructional material like traditional texts that can be reused in the composition of more complex objects like classes or courses. There are some difficulties in the process of LO reutilization. One of them is to find pieces of documents that can be used like LOs. In this work we present a process that, in search for LOs, starts by extracting, transforming and loading a text database and then continue clustering these texts, using a machine learning methods that combines EM (Expectation-Maximization) and a Bayesian classifier. We implemented that process in a system called "SQLLOMining" that uses the SQL language and text mining methods in the search for LOs.

Keywords

Web Based Education; Learning Objects; Ontology; Database; e-Learning; Machine Learning; Text Classification; Naive-Bayes; Internet.

Sumário

Lista de figuras

Lista de tabelas

O conhecimento é a pequena porção da ignorância que arrumamos e classificamos.

Ambrose Bierce

1. Introdução

O crescimento exponencial do conhecimento, maximizado nos anos recentes pelo advento da Internet e outros avanços da tecnologia da informação e comunicação, configura-se um grande paradoxo para o desenvolvimento do aprendizado humano, posto que, ao mesmo tempo em que se disponibiliza um vasto universo de informações pertinentes sobre praticamente quaisquer áreas de conhecimento, também se dificulta o acesso, pesquisa e processamento deste "excesso" de informação.

Esse paradoxo levanta questões importantes quanto à utilização e transferência desse conhecimento acumulado tais como: como organizar e integrar conhecimento a partir de um volume tão grande de informação desestruturada? Como gerir este conhecimento permitindo o seu reuso e aplicação de forma ágil e flexível? Como acessá-lo de forma rápida e prática?

Entendemos que boa parte das questões mencionadas anteriormente, como desafios importantes à gestão eficaz do conhecimento, são análogas aos problemas abordados e adequadamente endereçados durante anos de evolução constante da tecnologia de banco de dados. Sistemas Gerenciadores de Bancos de Dados (SGBDs) nada mais são que um conjunto de tecnologias que suportam armazenamento, manipulação, consulta e integração de bases de informação e, portanto, acreditamos que as mesmas abordagens utilizadas em bancos de dados são propensas a serem usadas no gerenciamento de conhecimento.

Todavia, algo que distancia bastante as técnicas hoje existentes para gerenciamento de bancos de dados daquelas necessárias à área da gestão de conhecimento é que as primeiras pressupõem que os dados estejam previamente estruturados (ex. Datawarehouse, DataMining etc), ou seja, os dados devem estar armazenados dentro de um padrão ou modelo.

Por outro lado, documentos existentes em bibliotecas encontram-se em sua maioria em formato de texto (ex. pdfs, docs etc), os quais constituem-se em uma base desestruturada e caótica, disponibilizados em uma forma totalmente despadronizada. Estes documentos possuem diversos elementos que poderiam ser utilizados na área de gestão de conhecimento, mas estes elementos precisam ser extraídos destes documentos e armazenados de uma forma estruturada, a fim de poder-se aplicar sobre os mesmos as técnicas de integração, consulta, e armazenamento que os bancos de dados disponibilizam.

Este trabalho faz parte do projeto Partnership in Global Learning (PGL) da PUC-RJ aonde um grupo de trabalho vem desenvolvendo estudos para usar a tecnologia de armazenamento de objetos de aprendizagem no desenvolvimento dos seus cursos em diversas áreas do conhecimento. Esta abordagem de armazenamento de LOs (objetos de aprendizagem) vem evoluindo na área de bancos de dados para e-learning (Melo e Baruque, 2003). Diversos trabalhos já foram desenvolvidos neste projeto todos voltados para a gerência de LOs como, por exemplo: (Baruque e Melo, 2005), (Gomes et al., 2006) e (Leal e Melo, 2006) .

O LO é descrito por elementos de metadados, a fim de proporcionar melhores buscas e reuso e eles podem ser constituídos de diversos ALOs (Atomic Learning Object) que são a menor unidade de aprendizado (Pereira, Porto e Melo, 2003). É preciso que os metadados possam ser definidos a partir do conteúdo existente no texto, permitindo o uso indiscriminado de qualquer conteúdo digitalizado em forma de ALO.

Propomos aqui a criação dos ALOs baseados em textos extraídos de documentos digitais. Sugerimos um processo semi-automático com este objetivo que faz uma seleção e classificação de partes destes documentos. Ao final deste processo um gestor de conhecimento ou o Instructional Designer pode fazer uma pesquisa definindo palavras de interesse e o tipo de ALO que deseja. A pesquisa retorna as partes dos textos, que possuem as palavras definidas além de uma classificação. O resultado é apresentado ordenado pela classificação feita de acordo com os parâmetros da consulta solicitada. Após a triagem final executada pelo gestor e baseado nos dados obtidos, fica mais simples definir um novo ALO que pode ser armazenado em um repositório de objetos de aprendizado de uma forma estruturada.

1.1.Motivação

Esta dissertação visa investigar métodos e técnicas para estruturação de conhecimento armazenado em textos. Quando consideramos a qualidade e a diversidade do conteúdo hoje existente em documentos digitais, antevemos um grande benefício na conjunção de técnicas de Bancos de Dados, Mineração de Textos, Aprendizado de Máquina e Gestão de Conhecimento para suporte a objetivos educacionais. Contribuir para o avanço, ainda que em parte, da extração e estruturação do conhecimento existente em milhares de documentos monolíticos existentes, e disponibilizá-los para utilização amigável por parte da comunidade científica e da sociedade em geral é um desafio muito motivante e objeto principal desta dissertação.

1.2.Objetivo

O objetivo principal deste trabalho é desenvolver um processo que permita a obtenção de partes de textos contidos em documentos que possam ser utilizados como Objetos de Aprendizagem. Investigamos métodos e técnicas que possibilitem e apóiem este processo.

Queremos que um usuário tenha como opção o tipo de texto desejado, o que especifica a proposta instrucional do texto que ele procura, e palavras chaves. Desta forma ele objetiva encontrar partes dos documentos que contenham um assunto de interesse em um formato específico, o que lhe resulta em um ALO. Como estas informações o gestor pode criar o ALO e preencher os dados existentes em seus metadados o que permite que eles sejam armazenados em repositórios de ALOs e utilizados na composição de LOs complexos e nas consultas integradas desenvolvidas em outros trabalhos para este tipo de repositório.

Utilizamos técnicas de análise da linguagem natural e Aprendizado de Máquina para gerar um modelo que permita a classificação de textos. Este modelo é então usado para ordenar os textos selecionados em um *ranking*, apresentando para o usuário no topo da lista os textos mais “parecidos” com o que foi solicitado.

Investigamos a utilidade e desempenho do Aprendizado Semi-Supervisionado para a execução da tarefa de classificação. Utilizamos o algoritmo proposto por (Nigam, Maccallum, Thrun e Mitchell, 2000) que, baseado na combinação de Expectation-Maximization (EM) e um classificador naive Bayes, aprende a classificação a partir de uma quantidade pequena de textos já classificados ou seja, exemplos que de alguma forma já foram etiquetados, acrescida de textos não etiquetados, que são mais fácil de serem obtidos.

1.3.Organização

No capitulo dois fazemos uma revisão sobre Objetos de Aprendizagem que é o que procuramos obter no processo de mineração de texto. Especificamos as classes que serão usadas no processo de classificação de textos e como elas foram definidas. No terceiro capitulo descrevemos as técnicas de Aprendizado de Maquina e todos os fundamentos teóricos utilizados no desenvolvimento do algoritmo de aprendizado e classificação e a justificativa para a utilização deste tipo de algoritmo de aprendizado. Definimos também as métricas utilizadas para medir a eficiência do classificador desenvolvido. No capítulo quatro apresentamos o Sistema SQLLOMining que foi desenvolvido para apoiar o processo de obtenção de LOs. No capítulo cinco apresentamos o Estudo de Caso detalhando os experimentos que foram feitos e os resultados alcançados. No capítulo seis apresentamos alguns trabalhos relacionados e faremos uma comparação. E finalmente no capítulo sete apresentamos as conclusões.

2. Objetos de Aprendizagem

Neste capítulo abordamos o objeto que almejamos obter a partir do processo de extração proposto.

2.1.LOs / ALOs

Existem muitas definições para objetos de aprendizado reutilizáveis (*Learning Objects* ou *Reusable Learning Objects* – LO ou RLO), mas existe um consenso em torno do conceito básico de porções reutilizáveis de conteúdo instrucional (Pereira, Porto e Melo, 2003). Algumas definições estão a seguir.

i. A IEEE Learning Technology Standards (IEEE Learning Technology Standards Committee, 2002) define um objeto de aprendizado ou objeto de aprendizado reutilizável como "Qualquer entidade digital ou não digital que possa ser usada, re-usada ou referenciada durante o aprendizado baseado em TI".

ii. David Wiley (Wiley, 2000) "Qualquer recurso digital que possa ser reusado para dar suporte ao aprendizado" ou "Qualquer coisa que possa ser entregue através da rede, grande ou pequena, sob demanda".

iii. O projeto da Cisco (CISCO, 1999) define "Um objeto de informação reutilizáveis (RIOs), ou seja, um "pedaço" de informação granular, reusável que é independente de mídia, que combinados formam estruturas maiores chamadas de *Reusable Learnig Object* (RLO). Enquanto o RLO é um "*wrapper*" que é colocado em torno de 7+-2 RIOs, cada RIO tem um objetivo de aprendizado específico que dá suporte ao objetivo geral do RLO.

Para o projeto PGL do TecBD utilizamos dois conceitos fundamentais que juntos dão origem ao LO: Aprendizado e Objeto. O lado Objeto traz as características comuns a este conceito na área de programação em linguagens

orientadas a objeto. E o lado Aprendizado nos remete ao tratamento que devemos dar ao conteúdo existente nestes objetos.

A orientação a objeto trouxe para o processo de modelagem de um conceito do mundo real, a criação de um objeto usado para representá-lo. As propriedades associadas ao objeto são representadas por atributos e funções que definirão a classe a que este objeto pertence. A definição da classe cria uma interface para a utilização de uma estrutura interna que pode ser desconhecida, o que chamamos de encapsulamento. A partir daí podemos trabalhar com os conceitos de herança e polimorfismo. A modelagem a objeto dá, claramente, a noção de especialização e generalização, o que nos permite relacionar classes de objetos dentro de uma hierarquia de herança. Todos estes conceitos nos trazem várias vantagens tais como abstração, agrupamentos e principalmente reutilização.

O modelo a objetos permite duas formas de reutilização. A primeira forma é a herança, que permite definir uma nova classe reutilizando classes já definidas. Este processo facilita a criação de novas classes permitindo focar apenas no que esta nova classe acrescenta. A segunda forma se utiliza do fato que cada classe é um pequeno módulo que pode ser reutilizado em outras situações.

Utilizando estes conceitos (Pereira, Porto e Melo, 2003) define que um LO pode ser composto por diversos *Atomic Learning Objects* (ALOs) que por sua vez pode ser definido como sendo o LO da classe mais elementar. Eles podem se agrupar de uma forma seqüencial ou não seqüencial criando LOs compostos. Estes LOs também podem se agrupar em novos LOs, o resultado final será um curso, ou um texto de um livro ou artigo que conterá diversas informações organizadas com o objetivo de transmitir um conhecimento, ou seja, com o objetivo de aprendizagem.

Após segmentarmos o objeto LO em diversos ALOs, poderemos facilmente reutilizar esta pequena porção de conhecimento, agora apreendida em um ALO, em diversas situações. Poderemos utilizar a definição de um termo específico em qualquer conteúdo que esteja tratando de algum conceito relacionado a este termo. Por exemplo, a definição de metro pode fazer parte de uma aula de física, onde está sendo apresentado o conceito de velocidade, assim como em uma

aula de metrologia, onde estão sendo apresentados os processos de medição existentes.

Podemos visualizar a formação de LOs da seguinte forma:

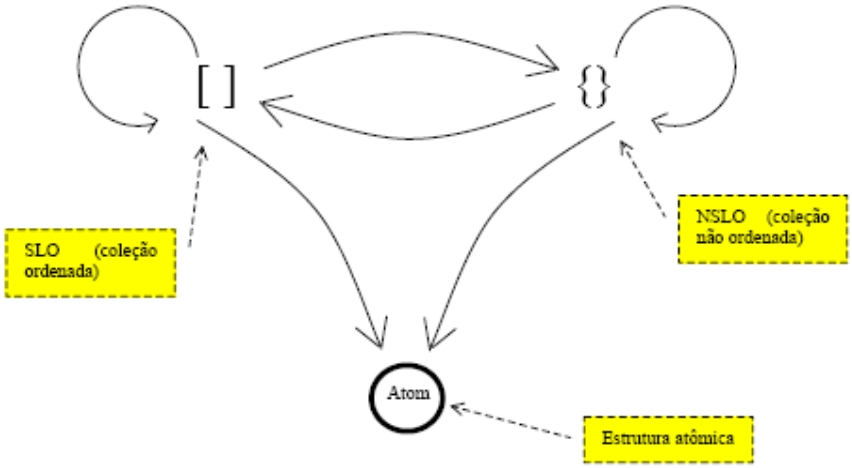

Figura 1- Modelo reprodutor de LOs (Pereira, Porto e Melo, 2003)

Onde SLO são grupos de LOs seqüenciais, ou seja, que possuem uma ordem específica de apresentação dos LOs mais elementares e NSLO são grupos de LOs não seqüenciais.

Cada ALO ou LO possui os seus metadados que o descrevem e serão utilizados no processo de integração e recuperação destes objetos visando à reutilização.

De acordo com a definição do W3C (W3C World Wide Web Consortium), metadados são informações localizadas na web, inteligíveis por um computador. Mais sinteticamente, podemos dizer que um metadado é um dado utilizado para descrever um dado primário. A importância dos metadados está basicamente ligada à facilidade de recuperação dos dados, uma vez que terão um significado e um valor bem definidos.

Quando criamos um modelo de classes utilizamos os metadados e os relacionamentos entre estas classes para acrescentar a semântica ao modelo. Estes modelos servem como um recurso que permite a especificação de um entendimento comum do que é um LO. Ele é o nosso vocabulário que especifica

o que estamos querendo encontrar. Estamos acrescentando semântica instrucional a um texto didático. Estamos especificando os possíveis objetivos instrucionais que um LO pode ter.

2.2.Definição das classes

Podemos encontrar hoje na literatura diversas propostas de como categorizar LOs. Todos estes trabalhos têm um grupo de classes de LOs e seus metadados. A seguir descrevemos um destes trabalhos que é muito referenciado pelo projeto PGL é o que possui mais especificações em relação ao conteúdo destes objetos.

Conforme (Cisco, 200) a Cisco adotou uma estratégia para criação de RIOs (Reusable Information Object), como eles denominam os ALOs, que especifica como transformar os seus cursos monolíticos e inflexíveis em objetos granulares e reutilizáveis que podem ser escritos independentemente da mídia e podem ser acessados dinamicamente através de um banco de dados. Estes RIOs contém itens de conteúdo, pratica e avaliação.

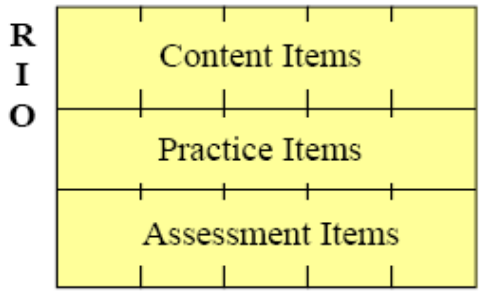

Figura 2 - RIO

E eles podem ser combinados para gerar um RLO (Reusable Learning Object). Este por sua vez possui uma introdução, um resumo, uma avaliação e cinco a nove (7+-2) RIOs.

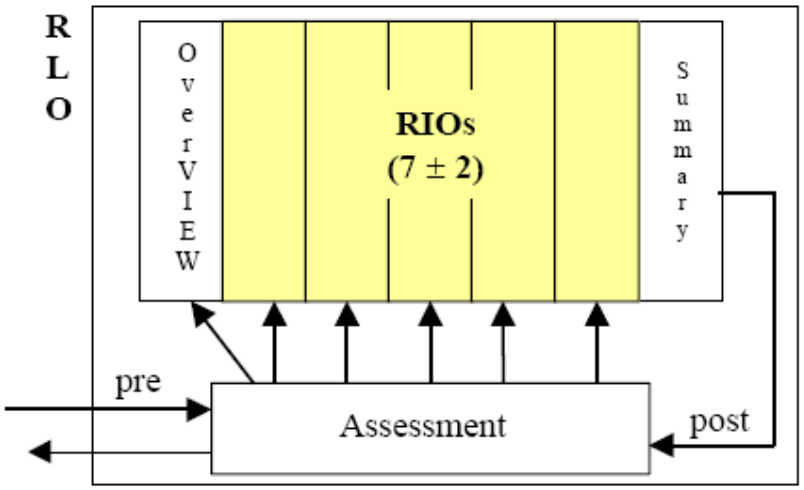

Figura 3 - RLO

Para definir uma metodologia de desenvolvimento de treinamentos criou-se uma estratégia de criação e categorização de conteúdos baseados em cinco tipos de informação: conceito, fato, processo, procedimento e princípio. Cada RIO é criado a partir de um único objetivo e todo o RIO pode ser classificado em um destes cinco tipos. Cada um destes cinco tipos é especificado e subdividido em alguns elementos.

Um RIO de conceito deve ser usado quando se quer ensinar um grupo de objetos, símbolos, idéias ou eventos que é designado por um único termo ou palavra, que compartilham uma característica comum, e que variam em características irrelevantes. O item de conteúdo de um conceito é composto por: introdução, definição, fato, exemplo, contra-exemplo, analogias e notas.

Um RIO de fato deve ser usado quando se quer ensinar um único e específico pedaço de informação. Um exemplo de fato é: “Este router tem quatro portas”. O item de conteúdo de um fato é composto por: introdução, ilustração, lista de fatos, tabela, notas.

Um RIO de procedimento deve ser usado quando se quer ensinar uma atividade a ser executada. Especificamente um procedimento é uma seqüência de passos que devem ser seguidos por um indivíduo para executar uma tarefa ou tomar uma decisão. O item conteúdo de um procedimento é composto por: introdução, fato, tabela de procedimento, tabela de decisão, tabela combinada, demonstração e notas.

Um RIO de processo deve ser usado quando se quer ensinar como um sistema funciona. Especificamente um processo é um fluxo de eventos que descreve como algo funciona. O item conteúdo de um processo é composto por: introdução, fato, tabela de estágios, diagrama de blocos, gráficos de ciclos e notas.

Um RIO de principio deve ser usado quando é necessário criar uma tarefa que requer algum julgamento ou quando direcionamentos precisam ser dados para uma tarefa. Princípio é composto por: introdução, fato, relação de princípios, direcionamentos, exemplos, não-exemplos, analogias e notas.

No metadados de um RIO encontramos diversos atributos e dentre eles podemos destacar: titulo, nível de objetivo, tipo, função, tarefa, nome do autor, nome do proprietário, data de criação, data de publicação ou data de validade e pré-requisitos.

Além deste foram criados diversos outros conjuntos de metadados que podem ser encontrados na literatura como, por exemplo, o Dublin Core e o LOM (Learning Object Metadata do IEEE LTSC) (IEEE Learning Technology Standards Committee, 2002). Alguns modelos desenvolvidos como, por exemplo, o SCORM e o CISCO RLO/RIO Model são analisados e comparados em (K. Verbert, E. Duval 2002).

2.3.Ontologia de tipos de ALOs

No artigo (K. Verbert, E. Duval 2002) os modelos de classes propostos para descrever os objetos de aprendizagem são avaliados e os limites impostos por este tipo de modelo são enumerados.

Nestes modelos a relação entre as classes é definida na sua estrutura, e estas relações se restringem a um tipo “é um (a)” (is-a). Por isso começou a se desenvolver ontologias, pois estas permitem que as relações sejam representadas fora da estrutura de classes e também abrem a possibilidade do uso de outros tipos de relação. Por exemplo, com uma ontologia podemos especificar que um LO “é” exemplo “para” (is-for) outro LO. Ontologias permitem

a descrição, inclusive a nível semântico, dos Objetos de Aprendizagem, o que virá a fortalecer o lado Aprendizagem do LO que faz parte dos conceitos fundamentais que juntos dão origem ao LO: Aprendizagem e Objeto.

A ontologia apresentada a seguir foi desenvolvida por Carsten Ullrich (Ullrich Carsten 2004, 2005) que diz que metadados como o LOM, apesar de fornecer um número abundante de propriedades, pecam em não representar uma informação de objetos de aprendizagem extremamente relevante que é a proposta instrucional daquele objeto. O atributo algumas vezes encontrado em metadados chamado de "Tipo" tenta resolver a questão, mas não tem o poder de descrever completamente esta parte do problema. Com esta ontologia, segundo Ullrich, é possível descrever o propósito do LO, o que facilitará em muito a reutilização deste objeto em novos Objetos de Aprendizagem.

A seguir descreveremos as classes e propriedades da ontologia de Objetos de Aprendizagem. A ontologia foi implementada utilizando Protégé (Gennari, et al. 2003) e está disponível em arquivo OWL[1]

Instructional Object. "Objeto de Aprendizagem" é a classe raiz da ontologia. Vários atributos são definidos neste nível: um identificador único "learning context", que descreve o contexto educacional para a audiência típica; e "field", que descreve a área desta audiência. Existe um espaço adicional para metadados Dublin Core.

Concept. A classe "Conceito" representa os LOs que descrevem a peça principal do conhecimento que está sendo transmitido. Conceitos puros são raramente encontrados em materiais didáticos. Na maior parte das vezes eles aparecem em uma forma especializada. Apesar disso, conceitos não são necessariamente voltados para um aprendizado específico, pois eles podem cobrir tipos de conhecimento em geral. Conceitos raramente são apresentados separadamente, normalmente eles dependem de outros conceitos. Isto está representado pela propriedade depends-on que pode ser preenchida com qualquer objeto da classe "Conceito".

1 - http://www.ags.uni-sb.de/~cullrich/oio/InstructionalObjects.owl

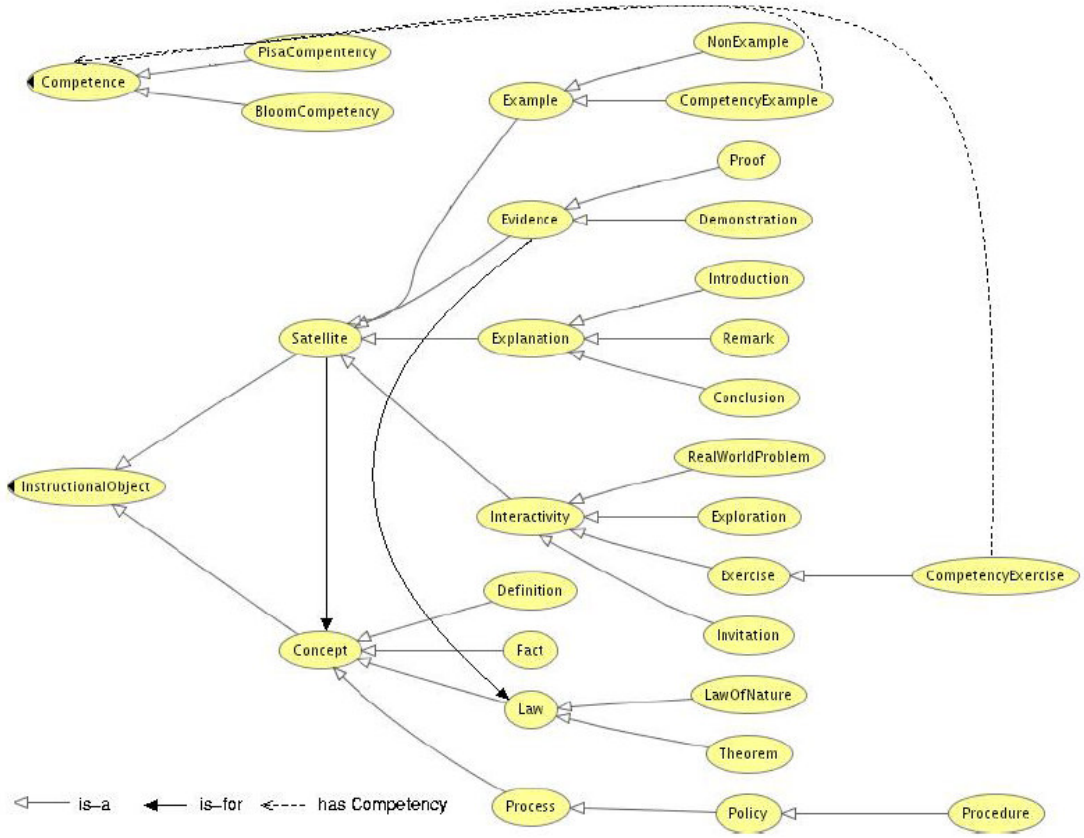

Figura 4 - Ontologia de Objetos de Aprendizagem (Ullrich Carsten 2004, 2005)

Fact. Um LO que é um "Fato" apresenta informações baseadas em ocorrências reais, eles descrevem um evento ou algo que é verdadeiro sem que seja uma regra. Um exemplo é a frase "Euclid viveu entre 365 e 300 BC".

Definition. Uma "Definição" é um LO que apresenta o significado de uma palavra, frase ou símbolo. Normalmente ele descreve um grupo de condições ou circunstancias que uma entidade deve preencher para que ela possa fazer parte de uma classe. Um exemplo é "A idade média descreve o período de tempo em que...".

Law. Um LO que é uma "Lei" descreve um princípio geral entre fenômenos ou expressões que foram provados ou são baseados em experiências consistentes.

Law of Nature. Uma "Lei da Natureza" é uma generalização científica baseada em observação. Um exemplo típico é: "Primeira lei de Kepler".

Theorem. Um “Teorema” é um LO que descreve uma idéia que já foi demonstrada como verdadeira. Em matemática, ele descreve uma sentença que pode ser provada como verdadeira, com base em premissas explicitas. Um exemplo é “A interseção de submonoids é um submonoid”.

Process. “Processo” e suas sub-classes descrevem uma seqüência de eventos. Quanto mais profundo na hierarquia de classes mais formal e especializado eles se tornam. Um processo provê informações em um fluxo de eventos que descrevem como algo funciona e podem envolver vários atores. Exemplos típicos são os processos digestivos ou como alguém é contratado em uma empresa.

Policy. Uma “Norma” descreve uma regra fixa ou predeterminada ou modo de agir. Um ator pode aplicá-la como um direcionamento informal de tarefas ou uma linha guia. Exemplo: Desenho de uma curva em matemática.

Procedure. Um “Procedimento” consiste em uma seqüência específica de passos ou instruções formais para alcançar um objetivo. Pode ser tão formal quanto um algoritmo. Exemplos típicos são: algoritmo de Euclid e instruções de como operar uma máquina.

Satellite. Estes elementos “Satelite” são LOs que apresentam informações adicionais sobre um conceito. A princípio conceitos fornecem todas as informações necessárias para descrever um domínio. Mas do ponto de vista instrucional os objetos “Satelite” contêm informações cruciais. Eles motivam o aprendiz, oferecem desafios e oportunidades de aprendizado. Todo o objeto “Satelite” oferece informações sobre um ou vários conceitos. Os identificadores destes conceitos são enumerados numa propriedade “for”.

Interactivity. Uma “Interatividade” é um LO que oferece aspectos interativos. Uma “Interatividade” é mais geral que um exercício, pois não possui necessariamente um objetivo específico que o aprendiz deve alcançar. Ele é desenhado para desenvolver ou treinar uma habilidade relativa a um conceito. A dificuldade de uma “Interatividade” é representada numa propriedade de mesmo nome. As sub-classes de “Interatividade” não capturam aspectos técnicos. Em

geral, o modo com a atividade é realizada, por exemplo, uma questão múltipla escolha, é independente da sua função instrucional.

Exploration. “Exploração” é um LO no qual o usuário pode explorar livremente os aspectos de um conceito sem um objetivo específico, ou com um objetivo, mas sem um caminho solução pré-definido. Simulações são exemplos típicos.

Real World Problem. “Problemas reais” são usados freqüentemente em LOs, especialmente em teorias construtivistas. Eles descrevem uma situação de vida cotidiana pessoal ou profissional que envolve questões abertas ou problemas.

Invitation. Um “Convite” é uma requisição para o aprendiz de executar uma atividade meta-cognitiva especifica. Por exemplo, pode ser uma chamada a discussão com outros estudantes.

Exercise. Um “Exercício” é um elemento interativo que requer uma resposta do aprendiz. A resposta pode ser avaliada (seja automaticamente ou manualmente) é uma nota pode se atribuída a ele.

CompetencyExercise. A classe “Exercício de Competência” permite que se especifique precisamente o objetivo educacional que um exercício requer que o estudante alcance quando o aprendiz pode aplicar um conceito.

Example. Um “Exemplo” serve para ilustrar um conceito. Da mesma forma que Interatividades, ele tem um campo que define a dificuldade.

CompetencyExample. Esta sub-classe de “Exemplo” é análoga a classe “Exercício de Competência”. Ele ilustra conceitos com objetivos educacionais diferentes.

Non-Example. Um “Não Exemplo” é um LO que não é um exemplo do conceito, mas é normalmente confundido como sendo. Nele incluimos contra-exemplos.

Evidence. Uma “Evidência” fornece observações ou provas de uma “Lei” ou uma de suas sub-classes. Desta forma a propriedade “for” de uma “Evidência” faz parte da classe “Lei”.

Proof. Uma “Prova” é uma evidência mais precisa. Ela pode ser um teste ou uma derivação formal de um conceito.

Demonstration. Uma “Demonstração” consiste de uma situação onde é mostrado que uma lei específica vale. Experimentos em física ou química são típicos exemplos de “Demonstração”.

Explanation. Uma “Explanação” provê informações adicionais sobre conceitos. Eles elaboram mais algum aspecto ou apontam claramente propriedades importantes.

Introduction. Uma “Introdução” contém informações que introduzem o conceito.

Conclusion. A “Conclusão” resume os pontos principais de um conceito.

Remark. Uma “Observação” provê informações adicionais não obrigatórias sobre um aspecto de um conceito. Pode conter lados interessantes ou detalhes de como o conceito está relacionado com outros conceitos.

Tendo descrito o que estamos à procura através de: (1) uso de conceitos de orientação a objeto, (2) criação de classes, (3) metadados e (4) uma ontologia de LO, podemos agora seguir com a descrição da proposta de como encontrar estes objetos. Utilizamos técnicas de aprendizado de máquina para apreender, a partir de exemplos, informações do que vem a ser o objeto que se quer encontrar, e esperamos que a ontologia apresentada facilite este processo de aprendizado.

3. Aprendizado de Máquina

O processo de pesquisa em aprendizado de máquina consiste em examinar e experimentar as estratégias mais eficazes para a construção de programas que aprendem a partir da experiência adquirindo conhecimento de forma automática. Um programa aprende um conjunto de tarefas *T* com uma medida de desempenho *D* a partir de uma experiência *E*, se seu desempenho de aprendizado *D* aumenta com a experiência *E*, ou seja, se é capaz de tomar decisões baseado em experiências acumuladas por meio da solução bem sucedida de problemas anteriores (Mitchell, 1997).

Existem diversas aplicações práticas deste processo em diversas áreas e um exemplo é a classificação de textos. A tarefa de classificação de textos se favorece em muito do aprendizado de máquina, pois esta técnica permite que o aprendizado ocorra apenas com a disponibilização de exemplos. Através da análise feita nestes exemplos o algoritmo consegue perceber o que diferencia um texto de outro e cria os parâmetros necessários que possibilitarão a classificação.

Então partimos de uma situação em que não é conhecida nenhuma relação à priori entre os dados de entrada e a saída desejada e, apenas através de exemplos, conseguimos criar esta relação.

O aprendizado de Máquina faz parte de uma área muito mais ampla chamada de Inteligência Artificial (Mitchell, 1997). Esta área procura estudar e compreender o fenômeno da inteligência e paralelamente desenvolve instrumentos para apoiar a inteligência humana. Outra área que apoia em muito o desenvolvimento de pesquisas em aprendizado de máquina é a Filosofia. Dela emprestamos metodologias básicas de desenvolvimento de conhecimento para aplicação em algoritmos de aprendizado de máquina.

A Filosofia é a primeira das ciências e foi ela que originou todas as outras existentes hoje. As metodologias desenvolvidas são muitas e dentre elas

podemos destacar o método de Aristóteles que consistia nas formas indutivas e dedutivas de se raciocinar. Basicamente o raciocínio dedutivo consiste em argumentar do geral para o particular, por exemplo: Todos os gatos miam, Mimi é um gato, logo Mimi mia. O indutivo por outro lado consiste em argumentar do particular para o geral, por exemplo: Mimi mia, Mimi é um gato, logo todos os gatos miam.

A dificuldade com o método dedutivo consiste na falta de premissas universalmente verdadeiras, pondo em cheque a eficácia do método de Aristóteles para descobrir a verdade. O raciocínio indutivo é mais característico do mundo moderno, pois está associado com a metodologia científica. Alguns opositores a este método argumentam que nunca se pode ter certeza de que se chegou a qualquer verdade através do método indutivo a não ser que se tenha observação completa ou universal, o que é impossível.

3.1.Tipos de Aprendizado

Utilizando o método de raciocínio indutivo existem vários tipos de aprendizado dentre eles: Aprendizado Supervisionado, Semi-Supervisionado e Não supervisionado. Para cada uma destas técnicas foram desenvolvidos diversos algoritmos. Para o aprendizado Supervisionado podemos encontrar: Transformation-Based Learning (TBL) e Naive Bayes. Para o Não supervisionado Hidden Markov Model (HMM). Alguns exemplos de algoritmos que utilizam o aprendizado semi-supervisionado são: EM (Expectation-Maximization) Modelo Bayesiano de Misturas Multinomiais e HMM. Muitos destes algoritmos possuem versões em mais de um tipo de aprendizado.

O aprendizado supervisionado utiliza pares associados de textos com seus atributos e a sua classificação, como exemplos para a criação de um modelo. No aprendizado não supervisionado, os exemplos não possuem valor de classificação associado e por isso são agrupados pelo algoritmo em clusters ou classes.

No aprendizado semi-supervisionado temos como objetivo aprender a classificação de textos utilizando um número limitado de exemplos etiquetados e acrescentar documentos não etiquetados. O processo de etiquetagem é

extremamente mais “caro” do que simplesmente a coleta de documentos não etiquetados.

Mas podemos nos perguntar como documentos não etiquetados (amostras) podem aumentar a precisão de uma classificação. Em (Nigam, Mccallum, Theun e Mitchell, 2000) muito é discutido sobre esta possibilidade. Este artigo diz que, a princípio, podemos pensar que não é possível obter nenhum ganho com isso. Mas as amostras efetivamente nos fornecem informações valiosas sobre a distribuição da probabilidade conjunta de n-gramas. Suponha, por exemplo, que utilizando apenas os exemplos podemos determinar que os documentos que possuam a palavra “é” são pertencentes à classe positiva. Se nós utilizarmos este fato para estimar a classificação de várias amostras podemos chegar à conclusão que a palavra “definição” ocorre freqüentemente nas amostras que agora se acredita serem da classe positiva. Esta coexistência das palavras “definição” e “é” numa quantidade muito grande de amostras pode nos fornecer informação muito útil na construção de um classificador mais preciso.

3.2.Algorítimo EM Bayesiano de Misturas Multinomiais

Neste trabalho utilizamos um algoritmo EM para treinar um classificador a partir de textos etiquetados e não etiquetados configurando um aprendizado do tipo semi-supervisionado, recurso extremamente útil quando temos acesso a poucos exemplos para o domínio do problema. Este algoritmo EM foi combinado conforme sugerido em (Nigam, Mccallum, Thrun e Mitchell, 2000) com o classificador naive-Bayes, um modelo de misturas de multinomiais que é bastante utilizado na tarefa de classificação de textos.

A seguir estaremos detalhando os conceitos envolvidos na definição deste Algoritmo EM Bayesiano de Misturas Multinomiais especificando cada uma de suas características.

3.2.1.Modelo Bayesiano

O modelo Bayesiano, assim como qualquer classificador probabilístico define um modelo probabilístico gerador para os dados e assume duas

premissas: (1) os dados são produzidos por um modelo de misturas e (2) existe uma correspondência de um para um entre as componentes desta mistura e as classes.

Desta forma todo o documento d_i é gerado de acordo com uma distribuição probabilística definida por um grupo de parâmetros denominado Θ. Esta distribuição consiste de uma mistura de componentes $c_j \in C = \{c_1,...,c_{|C|}\}$. Cada componente é parametrizada por um subconjunto de Θ. Desta forma um documento d_i é criado em dois passos: (1) selecionar uma componente da mistura de acordo com a sua probabilidade a priori $P(c_j \mid \Theta)$ e (2) gerar o documento através desta componente selecionada, de acordo com os seus parâmetros pela distribuição $P(d_i \mid c_j;\Theta)$. Então podemos caracterizar a probabilidade de um documento como a soma da probabilidade de todas as componentes da mistura.

$$P(d_i \mid \Theta) = \sum_{j=1}^{|C|} P(c_j \mid \Theta) P(d_i \mid c_j;\Theta) \quad (1)$$

3.2.2.Classificador naive-Bayes

Naive Bayes considera um modelo gerador probabilístico para textos. Este modelo é uma especialização do modelo de misturas descrito anteriormente e por isso assume as duas premissas já apresentadas. O termo ingênuo (naive) se referencia exatamente ao fato deste modelo adicionalmente assumir que existe uma independência entre os termos, ou seja, a distribuição conjunta dos termos é igual ao produtório da distribuição de cada um deles.

Um texto pode ser descrito pelo seu tamanho $|d_i|$ e por uma lista ordenada de termos $X = \{x_{i1}, x_{i2}...\}$. Então a componente selecionada gera uma seqüência de termos de tamanho especificado. Podemos então expressar a probabilidade de um documento dado a componente da mistura e Θ como sendo:

$$P(d_i \mid c_j;\Theta) = P(|d_i|)\prod_n P(x_{in} \mid c_j;\Theta;X)$$
$$P(d_i \mid c_j;\Theta) \propto \prod_n P(x_{in} \mid c_j;\Theta) \quad (2)$$

E chegamos a (2) utilizando a premissa naive-Bayes que os termos de um texto são gerados de forma independente do contexto, ou seja, independente dos outros termos existentes no mesmo texto. Isto quer dizer que assumimos que a probabilidade de um termo é independente da posição que ela aparece no documento. Assumimos também que para todas as classes o tamanho do documento é distribuído igualmente simplificando ainda mais o cálculo.

Assim, os parâmetros que definem cada uma das componentes associadas às classes é uma distribuição sobre os termos. Estas podem ser definidas utilizando várias distribuições tais como: a binomial ou a multinomial.

No modelo binário, assumimos que cada documento é representado por um vetor de atributos binários de modo que cada atributo indica a ocorrência ou não de um evento no documento. No modelo multinomial, assumimos que cada documento é representado por um vetor de atributos inteiros caracterizando o número de vezes que cada evento ocorre no documento. No nosso caso estaremos utilizando o modelo multinomial.

3.2.3. Distribuição Multinomial

A distribuição multinomial pode ser empregada na determinação da probabilidade quando, no evento especificado, se deseja calcular a probabilidade de um acontecimento composto, estabelecido por vários eventos. Neste caso, os eventos que constituem o acontecimento devem ser independentes e a ordem dos eventos, dentro do acontecimento, não influencia o cálculo da probabilidade.

Dado um grupo de variáveis $X_1, X_2, ..., X_n$ que possuem uma função probabilística:

$$P(X_1 = x_1, ..., X_n = x_n) = \frac{N!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} \phi_i^{x_i}$$

onde x_i são inteiros positivos tais que $\sum_{i=1}^{n} x_i = N$, e ϕ_i são constantes com $\phi_i > 0$ e $\sum_{i=1}^{n} \phi_i = 1$.

Então a distribuição conjunta de $X_1, X_2, ..., X_n$ é uma distribuição multinomial e $P(X_1 = x_1, ..., X_n = x_n)$ é dado pelo coeficiente correspondente da série multinomial $(\phi_1 + \phi_2 + ... + \phi_n)^N$.

Em outras palavras, se $X_1, X_2, ..., X_n$ são eventos mutuamente exclusivos com $P(X_1 = x_1) = \phi_1, ..., P(X_n = x_n) = \phi_n$. Então a probabilidade de X_1 ocorrer x_1 vezes, ..., X_n ocorrer x_n vezes é dada por (Papoulis, 1984):

$$P_N(x_1, x_2, ..., x_n) = \frac{N!}{x_1! ... x_n!} \phi_1^{x_1} ... \phi_n^{x_n} . \quad (3)$$

3.2.4.Modelo Bayesiano de Misturas Multinomiais

O modelo Bayesiano de Misturas Multinomiais então considera quatro premissas relativas à geração de documentos: (1) existe um modelo de misturas, (2) existe uma correspondência de um para um entre as componentes da mistura e as classes, (3) a ocorrência dos termos de um texto são eventos independentes e (4) o tamanho do documento e distribuído igualmente entre as classes. Todas estas premissas são violadas em textos reais: documentos podem comumente ser associados a mais de uma classe, os termos em um texto não são independentes, mas apesar disto empiricamente o classificador apresenta consistentemente bons resultados (McCallum e Nigam, 1998).

À medida que podemos estimar os parâmetros de Θ, obtendo θ, a partir dos documentos de treinamento, exemplos, é possível inverter o modelo gerador e calcular a probabilidade de um componente da mistura ter gerado um dado documento. Retiramos isto do teorema de Bayes e depois por substituição utilizando as eq. (1) e eq. (2) obtemos eq. (4).

$$P(c_j \mid d_i; \theta) = \frac{P(c_j \mid \theta) P(d_i \mid c_j; \theta)}{P(d_i \mid \theta)}$$

$$P(c_j \mid d_i; \theta) = \frac{P(c_j \mid \theta) \prod_n P(x_{in} \mid c_j; \theta)}{\sum_j P(c_j \mid \theta) \prod_n P(x_{in} \mid c_j; \theta)} \quad (4)$$

O documento será considerado pertencente à classe que tiver o maior valor calculado: $\arg\max_j P(c_j \mid d_i;\theta)$

Utilizando uma notação mais simples seguimos para as fórmulas que serão utilizadas no algoritmo.

$$\pi_j = P(c_j \mid \theta) \qquad \sum_j \pi_j = 1$$
$$\phi_{jp} = P(x_p \mid c_j;\theta) \qquad \sum_p \phi_{jp} = 1$$
$$\pi_{ij} = P(c_j \mid d_i)$$

Para um conjunto de textos etiquetados, o nosso conjunto de treinamento *D*, teremos $\pi_{ij} \in \{0,1\}$ e podemos afirmar a partir da eq. (1) que:

$$P(D \mid \theta) = \pi_1 P_1(D \mid \phi_{11},...,\phi_{1p}) + ... + \pi_j P_j(D \mid \phi_{j1},...,\phi_{jp}) \qquad (5)$$

E finalmente utilizando as distribuições multinomiais temos:

$$P_j(D_i \mid \theta) = \frac{N!}{n_1!...n_p!}\phi_{j1}^{n_1}...\phi_{jp}^{n_p} \qquad (6)$$

Quando estimamos θ queremos encontrar o Θ que maximize $P(\Theta \mid D)$. Utilizando o teorema de Bayes podemos quebrar a expressão $P(\Theta \mid D)$ em $P(D \mid \Theta)P(\Theta)$ e derivá-la em Θ. Este processo descrito em (Nigam, Mccallum, Thrun e Mitchell, 2000) nos levará às formulas que serão utilizadas no algoritmo EM (Expectation-Maximization) descrito a seguir para a estimativa de $\theta = (\pi_1, \pi_2,..., \pi_j, \phi_{11},..., \phi_{1p},..., \phi_{j1},..., \phi_{jp})$.

3.2.5.O Algoritmo EM

O algorítmo EM Expectation Maximization (Dempster, Laird e Rubin, 1977) é um processo iterativo eficiente para calcular a máxima verossimilhança estimada, quando existem dados faltando ou escondidos. No nosso caso as amostras são consideradas incompletas, pois a sua classificação não é utilizada para o aprendizado. No cálculo da máxima verossimilhança nós queremos estimar os parâmetros do modelo para os quais os dados observados são os

mais prováveis. A convergência é assegurada dado que o algoritmo garante aumentar a verossimilhança a cada iteração.

Cada iteração do algoritmo executa dois processos: o Expectation e o Maximization. No passo Expectation, ou E-step, os dados que estão faltando são estimados de acordo com os dados observados que geraram o modelo e seus parâmetros iniciais. No passo Maximization ou M-step, a verossimilhança é maximizada assumindo que os dados escondidos são conhecidos. A estimativa calculada para os dados faltantes no E-step são utilizadas para este fim.

Desta forma o E-step pode ser interpretado como sendo um passo construtor de um mínimo local para a distribuição a posteriori enquanto que o M-step aperfeiçoa este valor, melhorando a estimativa dos dados faltantes. Uma segunda referência sobre o Algoritmo EM é (McLachlan e Krishnan,1997).

O algoritmo EM foi combinado com o modelo Bayesiano de Misturas Multinomiais conforme sugerido em (Nigam, Maccallum, Thrun e Mitchell, 2000). Este processo permite que possamos utilizar documentos não etiquetados para a melhora do aprendizado diminuindo o trabalho manual na etiquetagem de exemplos. Estaremos assim transformando o nosso processo em um aprendizado semi-supervisionado. Note que a aproximação teórica utilizada depende das premissas já descritas: (1) os dados são produzidos por um modelo de misturas e (2) existe uma correspondência de um para um entre as componentes da mistura e as classes. Quando estas premissas não são satisfeitas, e na maioria dos casos reais isso acontece, aprimoramentos se fazem necessários como mostrado em (Nigam, Maccallum, Thrun e Mitchell, 2000). Em vários experimentos a adição de amostras, textos não etiquetados, piorou o desempenho do classificador.

A seguir descreveremos mais detalhadamente os passos do algoritmo EM e como ele calcula os parâmetros do modelo de misturas multinomiais:

3.2.6.Pseudocódigo

Faremos aqui uma breve descrição do processo como um todo incluindo cada um de seus passos. Aqui já utilizaremos a formalização próxima ao utilizado no algoritmo que foi implementado.

- Entrada: Conjunto de textos exemplo e textos amostra
- Criação de um modelo de mistruras multinomial inicial com a estimativa de Θ a partir apenas dos exemplos.
- Repetir até que não ocorra uma variação da melhora dos parâmetros maior que um fator especificado
    - E-Step – Utilizando o classificador atual Θ, estimar a probabilidade de cada documento amostra ter sido gerado por cada componente da mistura.

    - M-Step – Recriação do classificador Θ, utilizando os exemplos e amostras estimadas.
- Saída: Um classificador, que recebe um texto sem classificação e define a classe mais provável.

Para o E-Step temos o seguinte pseudo-código:

Loop i = 1 até a quantidade total de sentenças
Loop j = 1 até a quantidade total de classes

$$\pi_{ij} = \frac{\pi_j f_j(X_i \mid \phi_{j1},...,\phi_{jp})}{\sum_l \pi_l f_l(X_i \mid \phi_{l1},...,\phi_{lp})} \quad (7)$$

sendo que, para o cálculo de $f_j(X_i \mid \phi_{j1},...,\phi_{jp})$ fizemos algumas simplificações e utilizando a eq. (6) na eq. (7) obtemos:

$$\pi_{ij} = \frac{\pi_j \phi_{j1}^{n_{i1}} ... \phi_{jp}^{n_{ip}}}{\sum_l \pi_l \phi_{l1}^{n_{i1}} ... \phi_{lp}^{n_{ip}}} \quad (8)$$

E para o M-Step:

Loop j = 0 até a quantidade total de classes

$$\pi_j = \frac{\sum_i \pi_{ij}}{i} \quad (9)$$

Loop p = 0 até a quantidade total de palavras

$$\phi_{jp} = \frac{\sum_i \pi_{ij} n_{ip}}{\sum_l \sum_i \pi_{ij} n_{il}} \quad (10)$$

3.2.7.Suavização de Laplace

Com o propósito de evitar que o cálculo da probabilidade seja igual a zero pelo fato de uma das palavras não ocorrer em nenhum exemplo da categoria, podemos utilizar a suavização de Laplace que é muito utilizada em trabalhos encontrados na literatura.

Somamos então 1 ao numerador e o número de classes é somado no denominador no primeiro caso ficando assim:

Loop j = 0 até a quantidade total de classes

$$\pi_j = \frac{1 + \sum_i \pi_{ij}}{i + j} \quad (11)$$

Para o cálculo de ϕ somamos 1 no numerador e no denominador somamos o valor do tamanho do léxico obtendo:

Loop p = 0 até a quantidade total de palavras

$$\phi_{jp} = \frac{1 + \sum_i \pi_{ij} n_{ip}}{\sum_l \sum_i \pi_{ij} n_{il} + p} \quad (12)$$

3.2.8.Métricas *precision-recall* e F1

As métricas de $precision - recall$ são as mais utilizadas numa classificação binária. A tarefa de uma classificação binária muitas vezes se assemelha mais a uma filtragem do que a uma clusterização, pois encontramos poucos casos positivos em uma enorme quantidade de casos negativos. Imaginando a tarefa de obtenção de LOs podemos claramente observar esta situação. Em textos longos estamos interessados em separar apenas as partes que poderão ser utilizadas como objetos de aprendizagem. *Precision* e *recall* avaliam

corretamente esta tarefa e dada uma categoria c_i, $precision$ e $recall$ associados a esta categoria são definidos como:

Recall = Número de predições positivas corretas
Número de exemplos positivos

$$recall_i = \frac{TP_i}{N_i} \tag{13}$$

Precision = Número de predições positivas corretas
Número de predições positivas

$$precision_i = \frac{TP_i}{TP_i + FP_i} \tag{14}$$

Quando trabalhamos com diversas classes estamos calculando as mesmas métricas para cada classe separadamente e podemos sumarizar estes valores obtendo métricas globais da seguinte forma:

$$Macrorecall = \frac{\sum_i^{|C|} recall_i}{|C|} \qquad Macroprecision = \frac{\sum_i^{|C|} precision_i}{|C|} \tag{15}$$

Estas duas medidas globais apresentam normalmente resultados bem diferentes devido à distribuição de exemplos entre as classes. Caso esta distribuição seja muito disforme, o que ocorrerá com muita freqüência, as medidas por classe serão importantes assim como as medidas globais.

Para um classificador ser considerado bom não é suficiente que ele tenha uma destas medidas alta isoladamente. Por isso também calculamos uma terceira medida que faz uma avaliação conjunta delas da seguinte forma:

$$F_\beta = \frac{(\beta^2 + 1)\, precision \times recall}{\beta^2\, precision + recall} \tag{16}$$

Utilizamos β = 1 dando a mesma importância para as duas medidas e desta forma calculando a métrica chamada de F_1.

Adicionalmente em alguns experimentos utilizamos também a métrica Accuracy. Ela sozinha não é uma boa métrica de desempenho, pois poderíamos alcançar valores altos sempre classificando os textos na classe negativa, mas para os experimentos relativos ao aprendizado semi-supervisionado nos foi bastante útil. Esta métrica é na verdade a *precision* considerando todas as classes.

$$Accurrancy = \frac{TP}{TotaldeAmostras} \quad (17)$$

Finalizados os capítulos conceituais que apresentaram as bases teóricas do presente trabalho podemos seguir para a apresentação do processo que foi desenvolvido utilizando esta base. Apresentamos a seguir o sistema de mineração de LOs que foi desenvolvido para concretizar este processo.

4. SQLLOMining - um sistema de mineração de LOs

A extração de objetos de aprendizado de um texto através do aprendizado de máquina pode ser alcançada seguindo um processo que será desenvolvido e concretizado em um sistema que denominamos de SQLLOMining.

O nome escolhido faz referência a linguagem de programação utilizada Structured Query Language - SQL, o objeto de interesse LO e a técnica utilizada, mineração de textos. O Sistema SQLLOMining tem como objetivo prover uma ferramenta para a criação de ALOs para o projeto PGL a partir de textos existentes, utilizando técnicas de mineração de textos e aprendizado de maquina.

Este processo é composto de várias etapas. A primeira delas diz respeito à definição exata do que se procura, através da especificação de classes ou tipos de ALOs que serão o alvo do aprendizado de máquina. É necessária a criação de um Corpus de exemplos e amostras que deverá ser desenvolvido, a princípio, manualmente. Este Corpus nos permitirá não só a geração de um modelo inicial para o processo de aprendizado como também a classificação, com posterior avaliação, permitindo que possamos levantar os dados estatísticos que nos servirão de métricas do experimento feito.

Os arquivos que farão parte do Corpus deverão ser importados e associados a um tipo já especificado. Estes arquivos serão gravados no banco de dados e fragmentados em sentenças, que serão etiquetadas com o tipo definido para o arquivo.

A seguir apresentamos uma figura ilustrativa desta etapa inicial:

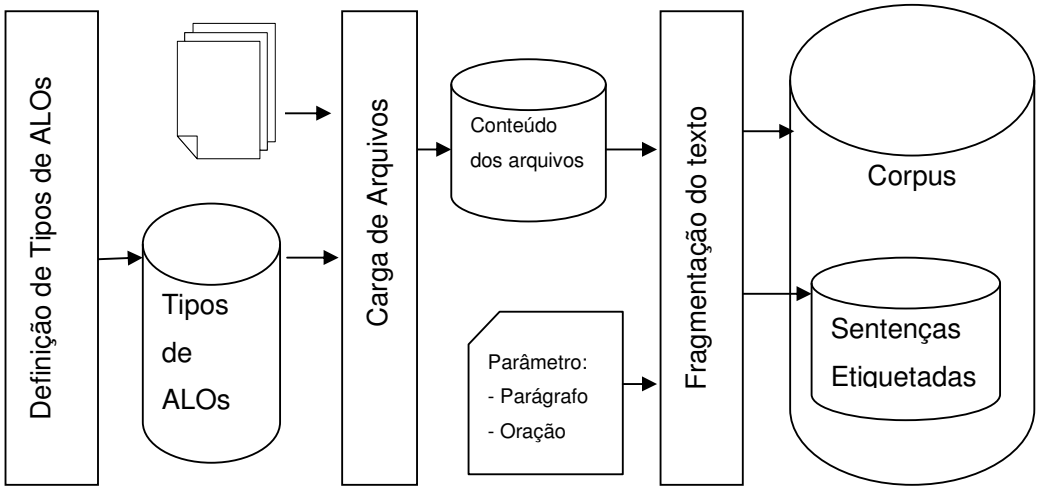

Figura 5 - Geração do Corpus Inicial

Na segunda etapa é feito um pré-processamento destes exemplos resultando na separação das sentenças do Corpus em n-gramas para utilização na análise e geração de input para o algoritmo de aprendizado. Poderemos então desenvolver o modelo propriamente dito, utilizando-nos de diversos recursos incluídos na ferramenta.

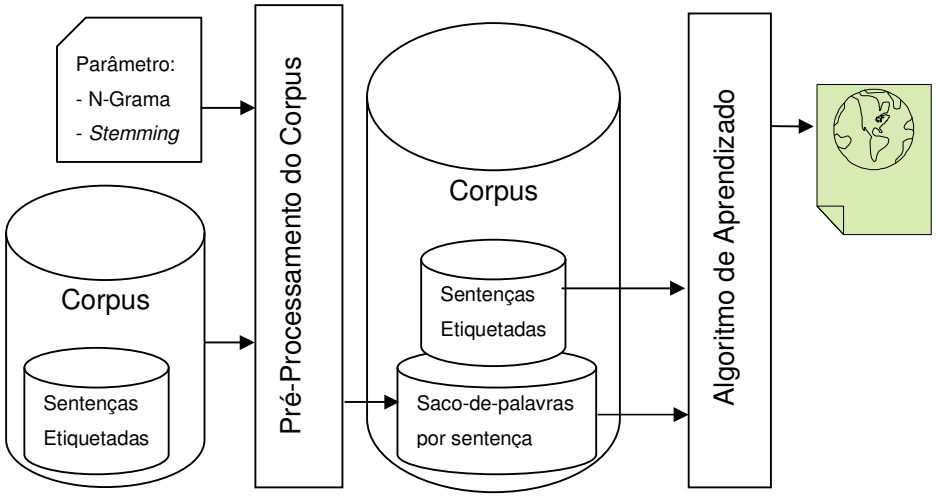

Figura 6 - Geração do Modelo Inicial

Por fim, uma vez gerado o modelo, poderemos utilizá-lo para classificação de qualquer texto apresentado pelo usuário, identificando claramente para ele as sentenças de seu interesse.

Para tanto, o usuário importará o novo arquivo a ser classificado, o qual sofrerá o mesmo processamento dos arquivos anteriores, e então será incluído no Corpus.

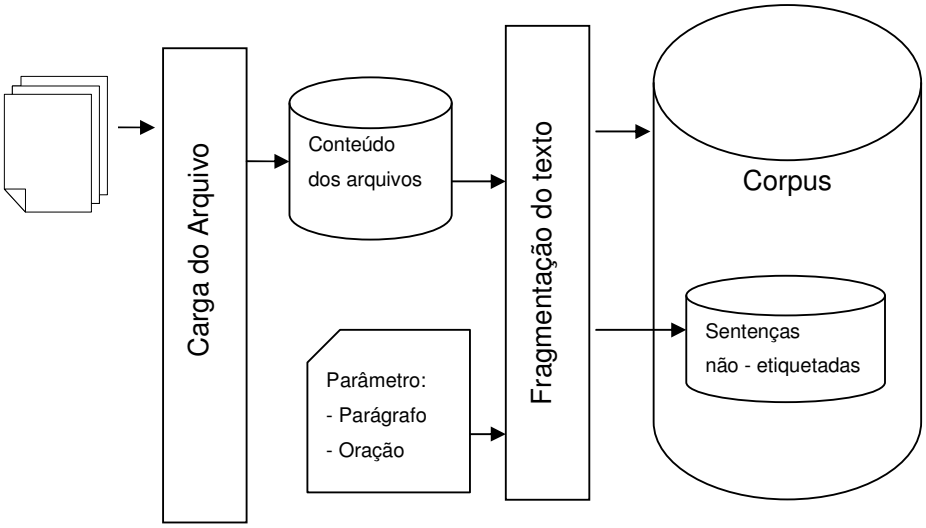

Figura 7 - Carga do Arquivo a ser classificado

Utilizando-se das sentenças não etiquetadas deste arquivo e do modelo inicial, o usuário poderá executar um aprendizado semi-supervisionado que classificará as sentenças não etiquetadas, aprimorando o modelo.

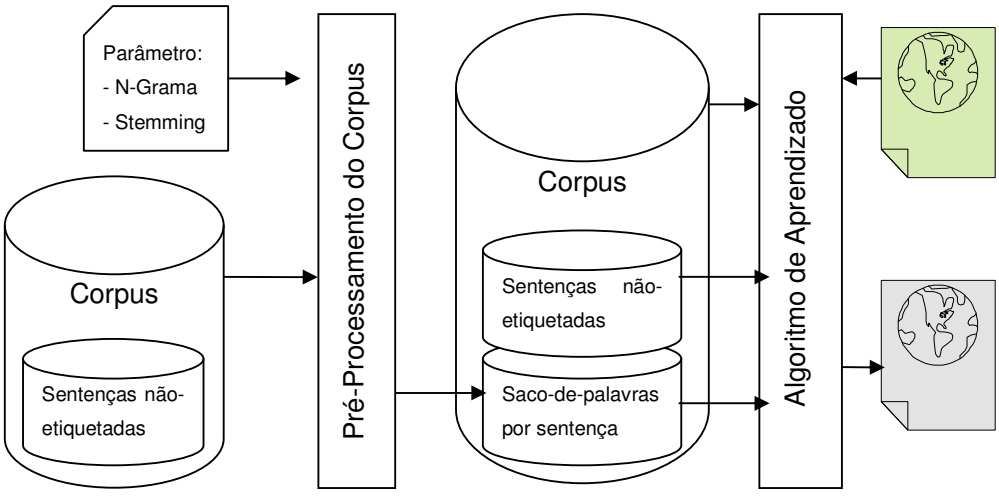

Figura 8 - Aprendizado Semi-Supervisionado

Finalmente, será possível através de telas de consulta selecionar dentre as sentenças pertencentes ao Corpus aquelas que forem de interesse do usuário.

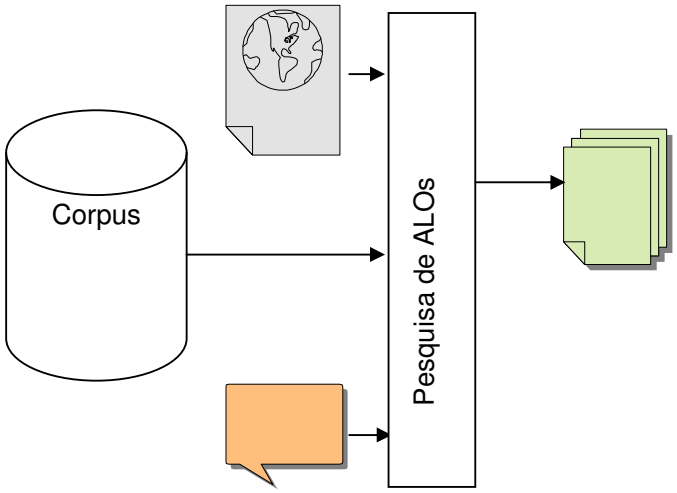

Figura 9 - Pesquisa de ALOs

4.1.Especificação do processo de extração de Objetos de Aprendizado

A seguir definimos mais claramente cada uma destas etapas e como elas foram disponibilizadas na ferramenta SQLLOMining.

4.1.1.Geração do Corpus Inicial

A palavra corpus significa corpo em latin. No contexto de Processamento de Linguagem Natural, corpus se refere a um conjunto de textos utilizados para experimentação e validação de modelos (Mitchell, 1997).

O Corpus inicial é gerado para apoiar a criação do modelo inicial. Faz parte do Corpus os tipos, ou classes, que o modelo irá utilizar, e arquivos contendo os exemplos. O Corpus inicial é gerado a partir destes exemplos, por isso todos os arquivos carregados para este Corpus deverão ser associados pelo usuário a um tipo de tal forma que todas as sentenças destes arquivos herdarão este tipo como etiqueta gerando um grupo de sentenças etiquetadas.

É possível utilizar diversos arquivos, cada um contendo exemplos de um dos tipos e diversos tipos. A definição dos tipos e destes arquivos permitem a

geração de experimentos diferentes, que dependerão do conteúdo deste Corpus Inicial. Os tipos definidos serão incluídos na tabela *classe* e os arquivos serão armazenados nas tabelas *Arquivostxt* e *ArquivosTxtConteudos.* Após a fragmentação do texto as tabelas *corpus* e *sentenca* serão preenchidas.

4.1.1.1.Definição dos tipos de ALOs

Disponibiliza-se para o usuário uma tela de cadastro de tipos. Nesta tela será informado apenas a Descrição ou nome do tipo e depois basta clicar no botão “Gravar”. A mesma tela pode ser utilizada para selecionar um dos tipos já cadastrados e então será possível alterar sua descrição ou exclui-lo. Para isso, depois de selecioná-lo basta clicar no botão “Limpar”.

Os tipos serão nada mais do que as classes que serão utilizadas no aprendizado de máquina. Poderemos trabalhar diversos experimentos variando a definição dos tipos. Desta forma poderemos investigar a combinação que melhor modelará o domínio de interesse. Lembramos que o modelo Bayesiano considera que cada texto está associado a apenas uma classe e por esta razão, a definição das classes é de extrema importância para o êxito da classificação.

Abaixo segue a tela do sistema associada a esta etapa do processo:

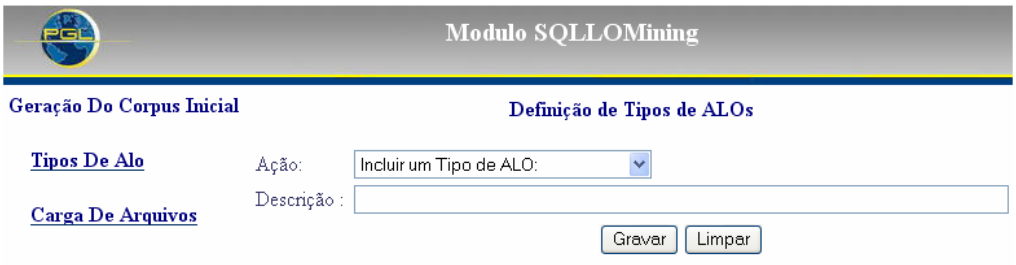

Figura 10 - Tela de Definição de Tipos de ALOs

4.1.1.2.Carga de arquivos

A Segunda tela apresentada abaixo é relativa ao processo de carga de arquivos. Na inclusão de um arquivo o nome dele deverá ser especificado e através do botão “Upload” o seu conteúdo será copiado para o banco de dados como *datatype image*. Desta forma será possível acessar o arquivo completo no

momento da consulta final. Após o preenchimento dos dados basta clicar no botão “Gravar”.

No processo de inclusão de um arquivo no banco, cada linha dele será lida e armazenada em uma linha de tabela. A seguir cada linha será desmembrada em palavras e em sentenças, sendo que as sentenças poderão compreender um parágrafo inteiro ou apenas uma oração. Estas sentenças serão distribuídas igualmente entre *testsets* que poderão depois ser manipulados separadamente, permitindo a utilização da validação cruzada na avaliação de métricas de desempenho do aprendizado de máquina.

Quando o arquivo for associado a um tipo, todas as sentenças que forem extraídas deles serão etiquetadas com o tipo associado. Será possível incluir mais de um arquivo por tipo ou também arquivos sem a associação de um tipo específico. Neste caso as sentenças apenas poderão ser utilizadas para classificação.

Nesta mesma tela será possível selecionar um arquivo já carregado para o banco para alteração ou para exclusão. O processo pode ser feito da mesma forma que na tela de Tipos utilizando os botões “Gravar” e “Limpar”. Abaixo segue a tela do sistema associada a esta etapa do processo:

Figura 11 - Tela Carga de Arquivos

4.1.1.3.Fragmentação do texto

Um ponto muito controverso é a fragmentação do texto importado em sentenças. Para nós existe uma unidade importante que definirá o conjunto de palavras que será analisado. Na realidade é possível encontrar uma variação muito grande em torno disso. Não existe um padrão que defina que um ALO deva possuir sempre apenas uma oração ou um parágrafo. Por isso disponibilizamos as duas alternativas que poderão ser escolhidas pelo usuário.

Na fragmentação por oração, o que definirá a quebra das sentenças é um "ponto final", "ponto de exclamação" ou "ponto de interrogação". Na segunda opção a quebra se dará apenas no parágrafo o que resultará em sentenças com várias orações. Com esta simplificação do problema estaremos perdendo informações valiosas, mas ganhamos em agilidade na obtenção de resultados. Clicando em um dos dois botões disponibilizados: "Fragm. por Oração" e "Fragm. por Paragrafo", a fragmentação do texto será executada.

Na mesma tela também serão disponibilizadas as sentenças resultantes do processo de desmembramento, para que sejam possíveis consultas após a fragmentação. As sentenças criadas serão apresentadas na tela uma a uma, utilizando para isso os botões "Próxima" e "Anterior". Durante esta consulta é possível alterar o tipo de uma sentença específica diferenciando-a das outras constantes no arquivo. Para isto basta selecionar a sentença e clicar no botão "Alterar". O botão "Limpar" exclui a sentença apresentada.

4.1.2.Geração do Modelo Inicial

A geração do Modelo Inicial é uma tarefa necessária e bastante importante no processo de classificação de textos. Este modelo servirá como base para o aprendizado semi-supervisionado e, como foi citado, será criado utilizando-se apenas os exemplos. Faz parte desta etapa do processo o pré-processamento do texto para a criação dos dados de input para o algoritmo e a geração do modelo multinomial propriamente dito.

O pré-processamento preencherá a tabela *Palavra* e, de acordo com os parâmetros informados, atualizará a tabela *Corpus*. Logo após preencherá a

tabela *Nip* que relacionará as palavras existentes em cada uma das sentenças e a quantidade de vezes que elas ocorrem.

Os dados do modelo serão armazenados nas tabelas *Plij*, fator de cada sentença para cada classe, e *Fljp*, fator de cada palavra para cada classe.

4.1.2.1.Pré-processamento do Corpus

O primeiro passo para a geração do modelo inicial é o Pré-processamento do Corpus. Após a extração do texto de arquivos txt para a base de dados, é necessário fazer um tratamento, via rotinas, facilitando o manuseio do conteúdo dos documentos. Estas rotinas permitem a estruturação da informação contida nos textos na forma de tabelas, de onde serão gerados os dados esperados para o input do algoritmo de aprendizado de máquina.

Abaixo relacionamos as técnicas de pré-processamento incluídas no SQLLOMining. Cada uma delas visa dar ênfase a alguma informação que está contida no texto, permitindo desta forma que ela não se perca quando a análise estatística for feita. Estas técnicas poderão ser utilizadas pelo usuário através de parâmetros de execução do pré-processamento, as quais serão disponibilizadas para preenchimento na tela correspondente.

Abaixo segue a tela do sistema associada a esta etapa do processo:

Figura 12 – Tela Pré-Processamento do Corpus

4.1.2.1.1.Saco de Palavras

Após a separação das sentenças é necessária a separação por palavra do texto para que seja possível a criação do saco de palavras (*Bag of Words*). Este modelo é bastante difundido nas aplicações de categorização, especificando quais palavras aparecem em cada sentença ou quantas vezes cada uma delas é utilizada. Este será o input principal para o algoritmo de classificação. No nosso caso estaremos utilizando a quantidade de ocorrência de cada palavra.

Constrói-se então uma tabela que conterá cada uma das palavras existentes nos textos analisados, além de uma outra tabela com a relação de palavras por sentença e a quantidade de vezes que aquela palavra aparece naquela sentença (*Nip*). A tabela abaixo ilustra o modelo de saco de palavras:

sentencaid	palavraid	Palavra	Nip
5546	78575	kelvin	1
5546	79164	Ação	1
5546	79258	Atuam	1
5546	80938	Contrários	1
5546	81024	e	2

Tabela 1 – Saco de Palavras Unigrama

4.1.2.1.2.N-Grama

Por se tratar apenas da contagem de vezes que a palavra aparece, diversas características do texto não são capturadas como, por exemplo, a ordem em que elas ocorrem. Isso dificulta a percepção da semântica do texto, pois diversas palavras na linguagem natural dependem do contexto para assumir significados distintos.

Um N-grama é um conjunto de N palavras consecutivas extraídas de uma sentença. O recurso de N-grama tem como objetivo capturar informações gramaticais no texto, aumentando a semântica capturada e amenizando a limitação do processo, devido ao fato de não se levar em conta a ordem em que as palavras são utilizadas nas sentenças.

O modelo de saco de palavras é levemente modificado, pois passamos a contar a quantidade de vezes que o conjunto de n palavras ocorre no texto. Um

caso em que tenhamos “é a” ou “é definido” é mais significativo do que apenas as palavras “é”, “a” e “definido” consideradas separadamente.

A tabela abaixo ilustra o modelo de N-Grama para N = 2. Este parâmetro estará disponível para o usuário escolher no momento da execução do passo de pré-processamento do texto. Caso o usuário queria utilizar o saco de palavras simples basta ele especificar o N-grama = 1.

sentencaid	palavraid	palavra	Nip
5546	78575	0 kelvin	1
5546	79164	ação e	1
5546	79258	atuam em	1
5546	80938	contrários e	1
5546	81024	corpos diferentes	1

Tabela 2 – Saco de Palavras Bigrama

4.1.2.1.3.Stemming

Freqüentemente, estamos observando diversas palavras que na verdade são somente variantes de uma única palavra. Plurais, formas de gerúndio e sufixos de tempo são exemplos de variações sintáticas, que impedem uma perfeita combinação entre uma palavra tratada como atributo e uma palavra do respectivo documento. Este problema pode ser parcialmente solucionado com a substituição de palavras pelos respectivos *stems* delas.

Stem é o conjunto de caracteres resultante de um procedimento de *stemming*. Ele não necessariamente é igual à raiz lingüística, mas servirá como uma denotação mínima não ambígua do termo. *Stemming* consiste em reduzir todas as palavras ao mesmo *stem*, por meio da retirada dos afixos da palavra, permanecendo apenas a sua raiz. O propósito é chegar a um *stem* que captura uma palavra com generalidade suficiente para permitir o sucesso na combinação de caracteres, mas sem perder muito detalhe e precisão. Um exemplo típico de um stem é “conect” que é o stem de “conectar”, “conectado” e “conectando”.

Em (Porter, 1980) é apresentado um dos métodos existentes, e que é o mais utilizado para este processo. A partir das linhas básicas deste método foi desenvolvido por Keith Lubell (Keith, 2006) um algoritmo em Structured Query Language (SQL). Utilizamos este algoritmo com algumas modificações sugeridas em (Matsubara e Monard, 2006) para a adaptação para a língua portuguesa. O

algoritmo desenvolvido numa *function* em SQL, e impresso como apêndice a este trabalho, utilizou três fases distintas onde cada uma delas tratava um tipo de pós-fixo. Estes tratamentos foram listados em uma tabela que foi também anexada para referência. Além disso, foram calculados os pontos de corte das palavras limitando a redução feita nas etapas de tratamento dos pós-fixos.

O usuário será convidado a definir um parâmetro que indicará se deverá ser utilizado *stemming* ou não. Caso ele defina este parâmetro como positivo o sistema executará o algoritmo *stemming* que gerará o *stem* de cada uma das palavras existentes no Corpus, e acrescentará esta informação a esta tabela (*Corpus*). Depois disso será criada a relação normal de palavras e também a relação de palavras por sentença baseadas no *stem* delas.

4.1.2.2. Algoritmo de aprendizado de máquina

Após a geração do Corpus devemos partir para o processo de criação do modelo que permitirá a classificação de textos. O algoritmo de aprendizado de máquina será utilizado para a geração do modelo a partir dos exemplos e para seu posterior aprimoramento através do mecanismo de aprendizado semi-supervisionado utilizando amostras adicionadas ao Corpus. A seguir falaremos sobre cada uma destas etapas do algoritmo de aprendizado de máquina.

4.1.2.2.1. Geração do Modelo Multinomial

Nesta etapa será possível especificar alguns parâmetros para a geração do modelo de misturas multinomiais. Os parâmetros dizem respeito ao *testset* a ser utilizado, além da especificação se será ou não utilizada a suavização de Laplace nos cálculos.

Nesta etapa é importante utilizar apenas sentenças etiquetadas ou seja, exemplos, permitindo a geração de um modelo o mais preciso possível. Como resultado, serão disponibilizadas na tela as medidas Accuracy e Abrangência alcançadas pelo modelo, quando aplicado aos mesmos exemplos que o geraram.

Na figura 13 segue a tela do sistema associada a esta etapa do processo:

Figura 13 – Tela Geração do Modelo Multinomial

4.1.2.3. Aprendizado Semi-Supervisionado

Num segundo passo, ainda desta etapa, será possível executar o aprendizado semi-supervisionado e acompanhar o resultado alcançado através das estatísticas calculadas a cada iteração. Neste passo os parâmetros solicitados serão, além dos mesmos da geração do modelo, a quantidade de amostras a ser adicionada.

O Aprendizado pode ser executado sobre um *testset* ou sobre um arquivo específico como detalhado no próximo item. Desta forma disponibilizaremos uma ferramenta bem poderosa, que permitirá o usuário classificar sentenças não etiquetadas e ao mesmo tempo aprimorar o seu modelo de Misturas Multinomiais.

Na figura 14 segue a tela do sistema associada a esta etapa do processo:

Figura 14 – Tela Aprendizado Semi_supervisionado

4.1.3.Carga do Arquivo a ser Classificado

Outra opção disponibilizada no sistema SQLLOMining é a carga de um arquivo do qual se deseja retirar os ALOs. Este procedimento terá algumas etapas que repetirão o que já vimos nos itens anteriores, mas executados apenas para um arquivo e não para o Corpus completo.

4.1.3.1.Carga e Fragmentação

O primeiro passo é carregar o arquivo para dentro do Corpus. Será possível definir um tipo para ele. A tela relativa a esta etapa segue abaixo.

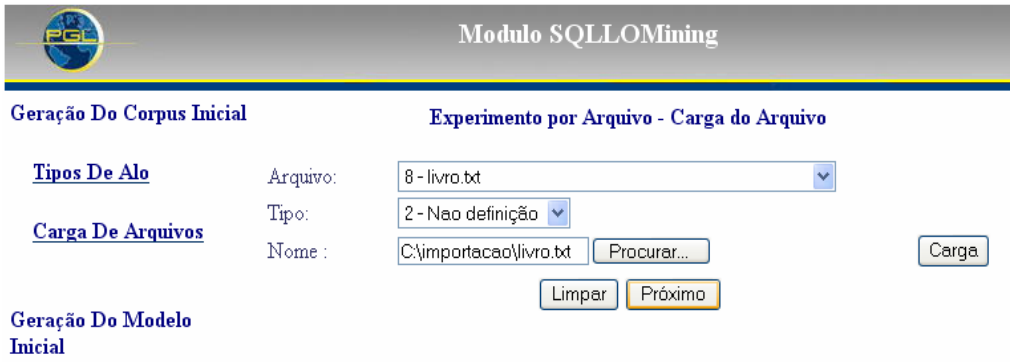

Figura 15 – Tela Experimento por Arquivo - Carga do Arquivo

Logo após, clicando no botão “Próximo” segue-se para o passo seguinte onde se pode escolher como se deseja fragmentar o texto existente neste arquivo. As sentenças resultantes poderão ser consultadas e o tipo de cada uma delas poderá ser alterado.

Figura 16 – Tela Experimento por Arquivo – Fragmentação do Texto

O próximo passo será o pré-processamento onde se especificará o n-grama e se o algoritmo de *stemming* será utilizado, da mesma forma que se fez no modelo. Deve-se sempre utilizar os mesmos parâmetros de forma que o modelo possa contribuir positivamente na classificação do arquivo que está sendo acrescentado ao Corpus.

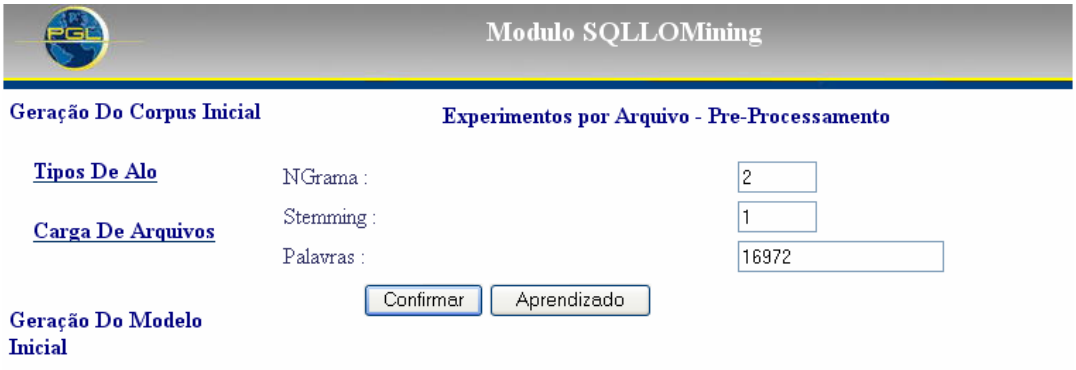

Figura 17 – Tela Experimento por Arquivo – Pré-rocessamento

4.1.3.2. Aprendizado Semi-supervisionado

O último passo será o aprendizado semi-supervisionado, onde o arquivo carregado será classificado e o modelo multinomial será recalculado utilizando-se o algoritmo EM já apresentado. Nesta etapa pode-se escolher se a suavização será utilizada ou não.

Figura 18 - Tela Experimento por Arquivo – Aprendizado Semi-Supervisionado

Como resultado, nesta mesma tela são apresentados os valores de métricas calculados após a classificação. Para cada execução de aprendizado estas métricas são recalculadas e listadas na tabela que se apresenta na tela. Clicando em uma das linhas da tabela principal o valor de *Precision* e *Recall* para cada classe é apresentado na tabela secundária, detalhando melhor os resultados alcançados.

4.1.4. Pesquisa de ALOs

A tela de pesquisa de ALOs fará a busca de acordo com os parâmetros: palavra chave e tipo, definidos pelo usuário. Na tela de resposta será disponibilizado tanto um *link* para o arquivo correspondente, como o trecho em questão. As sentenças selecionadas serão *rankeadas* pelos fatores calculados pelo classificador.

Abaixo segue a tela do sistema associada a esta etapa do processo:

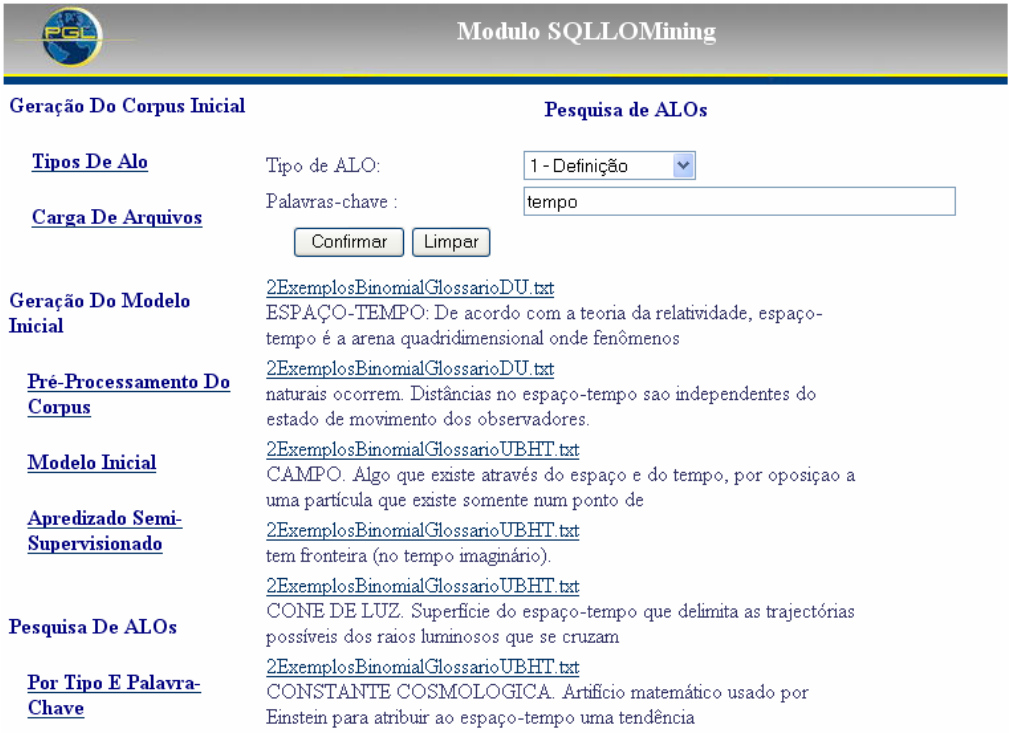

Figura 19 – Tela Pesquisa de ALOs

Acrescentamos também a pesquisa de sentenças classificadas erradas, acrescida de dois parâmetros: o tipo e palavras-chave de interesse. Ela retornará as sentenças classificadas em uma classe específica, que é definida como um dos parâmetros da consulta, e que é diferente da definida na etiquetagem da sentença. E a última pesquisa fornecida seleciona as sentenças pertencentes a um arquivo especificado, e que tenham sido classificadas em um tipo, também especificado pelo usuário. Ela retornará as sentenças que foram identificadas pelo classificador como ALOs, e que são pertencentes a um dos arquivos carregados para classificação. Nesta pesquisa também é possível definir como parâmetro palavras-chave de interesse.

Estas consultas tornaram-se úteis durante os experimentos feitos, pois elas permitiram a avaliação dos resultados alcançados e a análise de casos específicos. Muitas vezes estas análises puderam orientar os experimentos de modelagem, indicando ocorrências especiais na classificação de sentenças que possuíam uma característica singular no texto.

Abaixo segue a tela do sistema associada a esta etapa do processo:

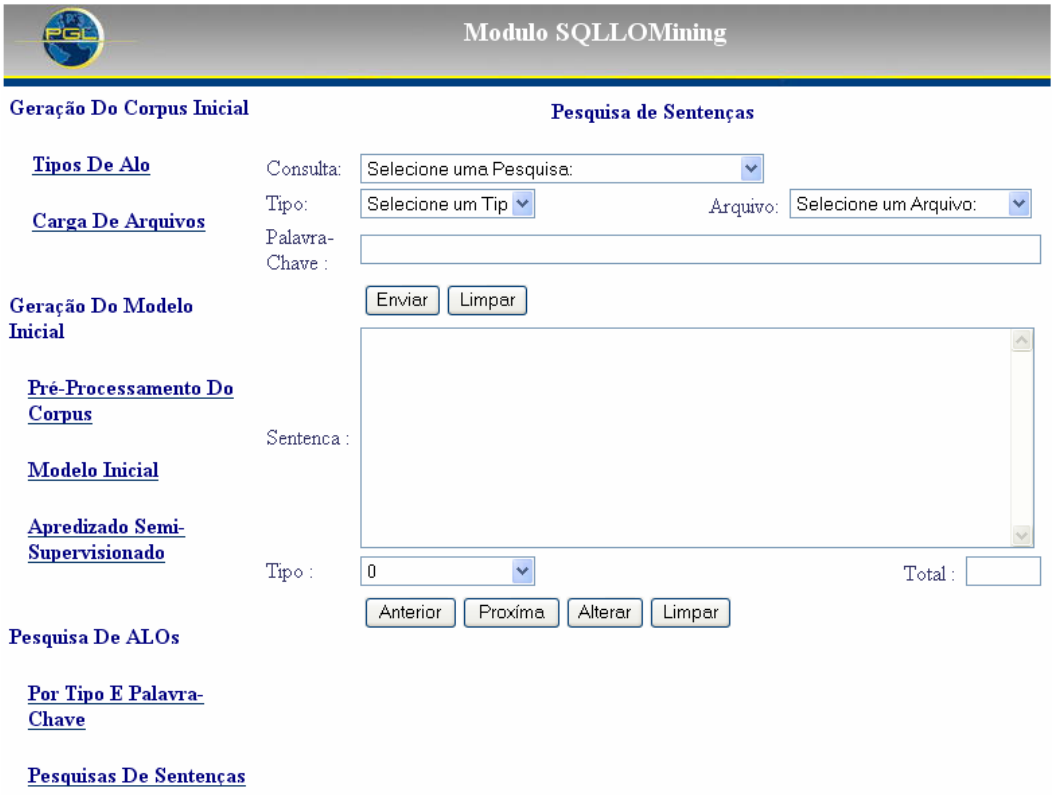

Figura 20 – Tela Pesquisa de Sentenças

4.2. Diagramas de Especificação

A seguir faremos uma especificação formal do sistema desenvolvido com diagramas de classe e de entidade relacionamento. O sistema foi desenvolvido na plataforma Java utilizando o Framework Struts (Struts) e um banco de dados relacional.

Alguns dos motivos para utilização do Strutus Framework que podemos enumerar são: garantia da Apache Group que manterá a manutenção e aprimoramento do framework (correção de *bugs* e novos *releases*); foco de esforços em regras de negócio; separação da camada de negócio da camada de apresentação; criação de aplicações padronizadas, facilitando a manutenção; criação de aplicações internacionalizadas; possibilidade de gerar a saída de acordo com o dispositivo usado (HTML, SHTML, WML, etc). A Struts está disponível sobre a licença "*free-to-use-license*" da Apache Software Foundation.

Todas as rotinas de cálculo e pré-processamento de texto foram feitas em Structured Query Language - SQL. Elas estão disponibilizadas no Apêndice II deste documento.

4.2.1. Diagrama de Classes

Abaixo segue o diagrama de classes com a estrutura utilizada na programação em java. Esta parte do sistema diz respeito apenas à manipulação de informações básicas, que disponibilizará as telas de cadastro e de consulta descritas anteriormente. A parte maior do sistema foi implementada em SQL utilizando Procedures, que são comentadas mais a frente.

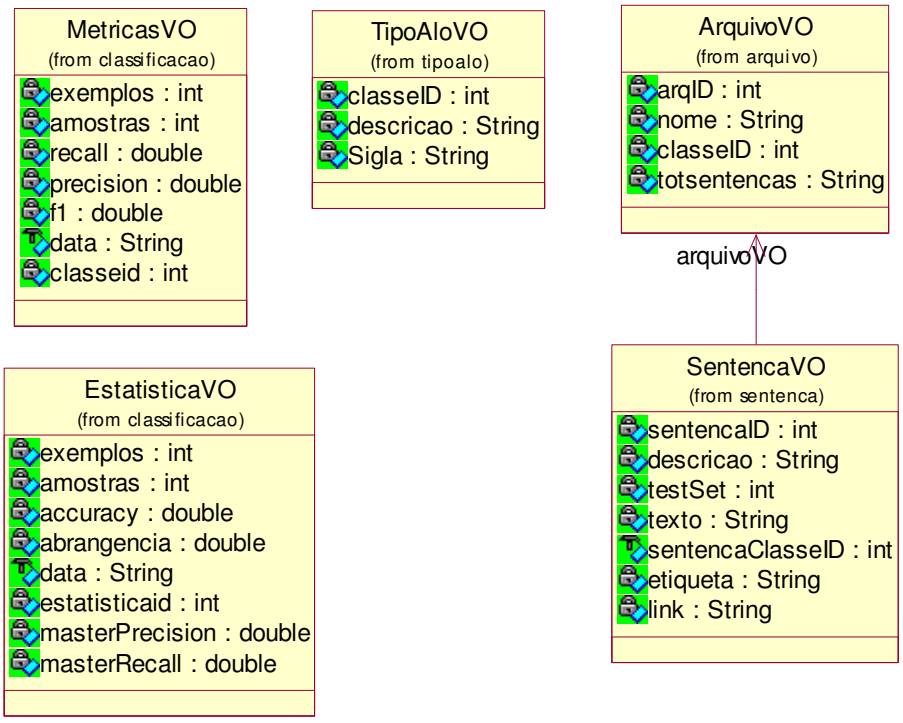

Figura 21 - Diagrama de Classes

Neste digrama apresentamos apenas as classes que representam os tipos de ALOs que o usuário cadastrou, os arquivos que foram carregados pelo usuário e as sentenças que foram geradas a partir da fragmentação destes arquivos. Adicionalmente temos as classes que representam os resultados obtidos e que são retornados para usuário, apresentando as métricas do desempenho obtido na classificação.

4.2.2.Diagramas de Modelagem

O diagrama Entidade Relacionamento da figura abaixo apresenta as entidades principais e os seus relacionamentos.

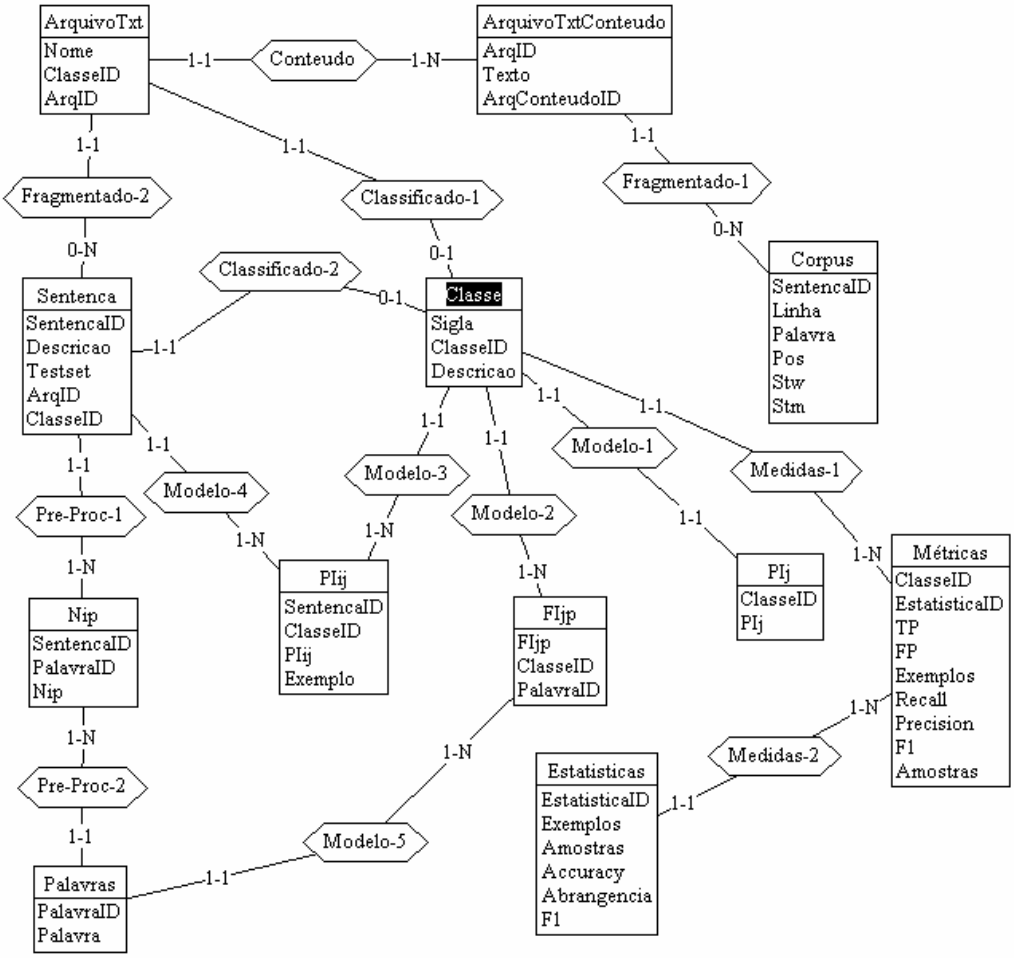

Figura 22- Diagrama ER

No primeiro passo do processo as entidades *Classe*, *ArquivosTxt* e *ArquivosTxtConteudo* são criadas com os dados informados nas interfaces de cadastro disponibilizadas para os usuários. Feito a fragmentação do texto as entidades *Corpus* e *Sentenca* são criadas com as partes dos textos. Na entidade *Corpus* a separação é feita ao nível de palavra e as sentenças são separadas e listadas na entidade *Sentenca*.

No segundo passo do processo é feito o Pré-processamento do Corpus. Neste momento as entidades *Palavras* e *Nip* são geradas, a primeira com a lista

de palavras a serem usadas na modelagem, ou seja, com a lista das *features* do modelo e a segunda com o saco-de-palavras por sentença onde são listadas as palavras que ocorrem em cada sentença com a quantidade de vezes que cada uma delas aparecem.

No terceiro passo o algoritmo será utilizado para a geração do modelo. Neste momento os parâmetros estimados do modelo serão calculados e gerados em *Plj* e *Fljp*. E finalmente no último passo faremos o aprendizado semi-supervisionado. Inicialmente as amostras serão estimadas com os valores de *Plij* e depois o modelo será reestimado alterando os valores mencionados anteriormente.

Para estas etapas do algoritmo o foram implementadas três procedures: ModeloAPartirDeExemplos, ChuteInicialAmostras e AprendizadoAmostras, elas estão disponíveis no Apêndice.

É interessante ressaltar que todos os cálculos de somatórios foram facilmente implementados na linguagem SQL, pois podem ser executados em apenas um comando. Como exemplo temos a fórmula (9) apresentada no capítulo 4 que especifica o cálculo para um dos parâmetros do modelo.

Loop j = 0 até a quantidade total de classes

$$\pi_j = \frac{\sum_i \pi_{ij}}{i} \qquad (9)$$

Ao invés da execução de uma repetição é necessário apenas fazer um somatório da forma apresentada a seguir, utilizando a estrutura do banco de dados para a execução do cálculo em apenas um comando.

```
INSERT INTO Plj (classeID, Plj)
SELECT classeID, SUM(Plij) / (SELECT COUNT(*) FROM Plij)
FROM Plij
 GROUP BY classeID
```

A seguir apresentamos o modelo lógico do banco de dados do sistema SQLLOMining.

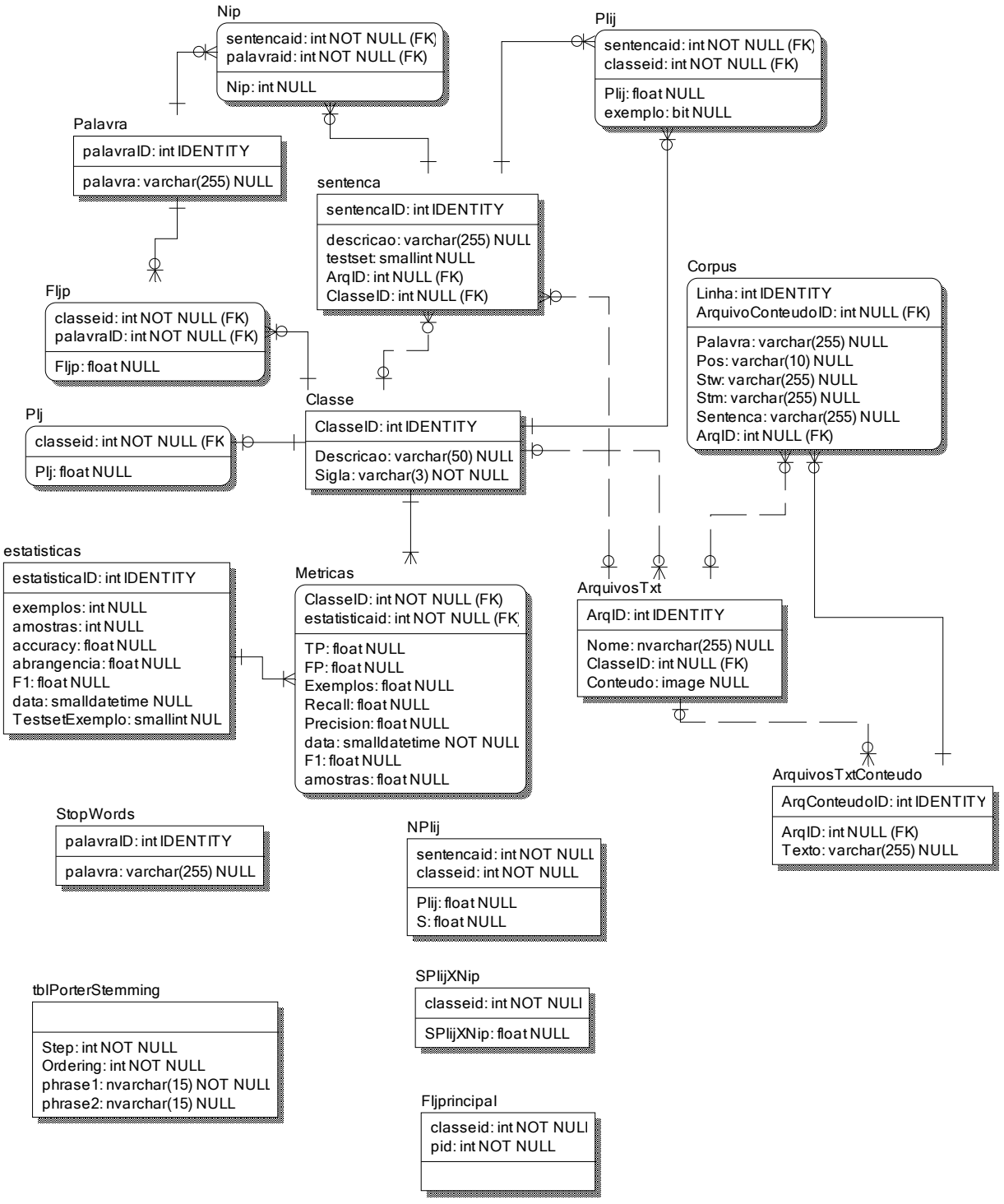

Figura 233- Modelo Lógico

Estão apresentadas neste diagrama todas as entidades mencionadas anteriormente em forma de tabelas já com suas colunas e seus relacionamentos.

5. Estudo de Casos

Neste capítulo apresentaremos os experimentos feitos com a ferramenta SQLLOMining objetivando a obtenção de Objetos de Aprendizagem. Descreveremos também alguns procedimentos adotados para a geração dos dados que foram utilizados nos experimentos e finalmente apresentaremos os resultados obtidos.

5.1. Geração do Corpus

O processo de especificação do Corpus foi um trabalho que partiu praticamente do zero. Em alguns domínios de problemas mais comuns, já existem Corpus disponibilizados em um formato padrão, mas no estudo de caso que nos propomos trabalhar, encontramos um material disponível extremamente amplo e rico, mas totalmente desprovido de qualquer tratamento. O nosso material consiste de qualquer documento que possa ser encontrado na Internet ou em uma biblioteca digital que possua o conhecimento que queremos capturar em forma de texto.

Para restringir um pouco esta vasta opção de recursos, optamos por nos concentrar em textos de um determinado domínio permitindo assim o desenvolvimento de um Corpus especializado e por isso mais rico em informação para o aprendizado de máquina.

Apesar de estarmos restringindo o nosso domínio, estamos falando sempre de extração de conhecimento através do aprendizado a partir de exemplos. Isso nos leva a uma necessidade de exemplos extremamente precisos e ricos em informação. Eles precisam espelhar claramente o objeto de interesse representando da melhor forma as classes de texto que podem ser encontradas. Desta forma estaremos procurando manter o modelo o mais próximo possível das premissas adotadas na sua formulação e assim reduzir erros de classificação. Para isso utilizaremos a ontologia de tipos de ALOs descrita anteriormente que visa descrever o domínio que estamos abordando.

Outro resultado positivo que podemos esperar obter com a restrição de um domínio é a restrição também do léxico e da forma de escrita. O uso da linguagem natural pode ser extremamente variado e esta variação ocorre por diversas causas seja pelo lado de quem está escrevendo como pelo lado de para quem o texto está sendo escrito.

Um texto escrito por um romancista é totalmente diferente do que um escrito por um mestre ou doutor. E da mesma forma, um texto escrito para alunos do ensino médio tem uma abordagem totalmente deferente de um escrito para aqueles mesmos mestres ou doutores.

Partimos então das seguintes premissas: trabalharemos com um tipo de texto específico, voltado para o ensino médio, que apresentem conceitos de áreas específicas como física e química e procuraremos nestes textos parágrafos que possam ser classificados na ontologia de tipos de ALOs e utilizados na criação de Objetos de Aprendizagem (LOs).

5.1.1.Extração automática de exemplos

O processo de coleta de exemplos é extremamente trabalhoso. A quantidade mínima requerida para começarmos a alcançar resultados razoavelmente satisfatórios é grande, e fazer uma etiquetagem manual é praticamente impossível quando se quer alcançar este número. Partimos então para uma abordagem mais simplista que nos permitisse desenvolver um Corpus razoavelmente satisfatório.

Uma das abordagens objetivou a busca de textos que contivessem apenas um tipo de sentença. Um exemplo muito fácil de encontrar são textos de definição. Um glossário, que pode ser facilmente encontrado na Internet, possui apenas textos da classe Definição e nos permite colecionar de uma vez centenas de exemplos válidos.

Por outro lado, textos romanceados não possuem definições e podem ser considerados como contendo apenas sentenças da classe negativa com uma margem bastante pequena de erro.

Partindo destas abordagens colecionamos mais de 300 exemplos de definições do domínio de física extraindo glossários de termos de física e química disponíveis na Internet e utilizamos como exemplos da classe negativa, textos de livros que falavam sobre o mesmo assunto, mas de uma forma romanceada.

5.1.2.Pré-etiquetagem com conferência manual

Outra abordagem válida e que nos permitiu gerar uma quantidade de exemplos bastante satisfatória é a pré-etiquetagem com conferência manual. Este processo necessita de um modelo já criado a partir de exemplos já coletados. Este modelo é utilizado para etiquetar textos ainda não classificados e sobre esta etiquetagem é feita uma conferência revisando apenas os textos etiquetados positivamente.

Este método nasceu naturalmente da técnica de aprendizado semi-supervisionado utilizada pelo algoritmo existente no sistema SQLLOMining onde na segunda fase do aprendizado o algoritmo classifica as amostras a partir do modelo pré gerado e aprimora seus parâmetros estatísticos aumentando sua precisão.

5.2.Descrição dos experimentos

A seguir apresentaremos os dois Corpus que foram desenvolvidos utilizando a ferramenta SQLLOMining. Ambos utilizaram textos orientados ao ensino médio de ciências exatas como, por exemplo, a física e a química.

5.2.1.Corpus Definição

O primeiro Corpus que desenvolvemos possui apenas dois tipos sendo um positivo e um negativo. Estamos objetivando encontrar apenas textos relativos a definições de conceitos separando-os de todo o resto. Este Corpus utilizou a técnica descrita anteriormente de extração automática de exemplos através de glossários de termos de física encontrados na Internet e em livros da área.

Para exemplos de texto da classe não-definição, utilizamos livros romanceados relativos ao mesmo assunto como, por exemplo, “Uma Breve

História do Tempo" de Stephen Hawking (Hawking, 1988). Criamos então o Corpus Inicial.

Numa segunda etapa fizemos uma revisão manual das sentenças dos livros que foram consideradas como da classe positiva pelo algoritmo e encontramos diversas delas que eram realmente definições. Um exemplo é a seguinte sentença: "Com efeito, o metro é definido como a distância percorrida pela luz em 0,000000003335640952 segundos medidos por um relógio de césio." Ela está no livro "Uma Breve História do Tempo" e inicialmente estava definida como sendo da classe negativa. Corrigimos então a classificação inicial delas as incluindo como exemplos da classe positiva.

Acrescentamos o livro "A Dança do Universo" de Marcelo Gleiser (Gleiser, 2006) e refizemos este procedimento utilizando todos os tipos de modelagens possíveis variando o parâmetro de N-Grama e *stemming*. Aumentamos com isso a quantidade de exemplos de definição que tínhamos inicialmente. Alcançamos então o número de 628 exemplos de definição e 2558 exemplos de não-definição para o livro "Uma Breve História do Tempo" e "A Dança do Universo". Chamamos este Corpus de Corpus Aumentado.

Fizemos os experimentos de aprendizado com o Corpus desenvolvido nas etapas anteriores com o objetivo de colher as métricas de desempenho do aprendizado que serão apresentadas a seguir. Para executar os cálculos destas métricas consideramos todo o resto de amostras provenientes dos livros como sendo da classe negativa. Esta é uma premissa simplista, mas consideramos uma aproximação razoável devido ao fato de realmente serem muito reduzidos os casos de definição existentes no texto dos livros.

Devido a quantidade reduzida de exemplos utilizamos apenas dois grupos de teste e por isso executamos os experimentos fazendo uma validação cruzada reduzida que utilizou estes dois grupos de exemplos.

5.2.2.Corpus Estendido

Como mencionado em (Chakrabarti, 2002) o modelo Bayesiano de misturas parte do pressuposto que cada texto deve estar relacionado a uma e apenas uma classe assim como cada componente da mistura. Esta mistura

representará a geração de todo o documento existente. Partindo destas premissas podemos supor que, para que possamos obter um modelo que possa classificar qualquer texto, ele deve estar munido de informação suficiente para conseguir classificá-lo corretamente em uma das classes que ele contempla. Por isso seria importante que este modelo estivesse "informado" em relação a todos os tipos de texto que existem.

Sob este ponto de vista procuramos sempre trabalhar com um conjunto de classes que satisfaça essas premissas e para definir este conjunto utilizamos a ontologia apresentada anteriormente. Como a tarefa de classificar qualquer texto que existe é bastante complexa, usamos continuamente a classe negativa. Esta classe engloba todos os outros tipos de texto que não têm uma classe especificada. Por exemplo, nos primeiros experimentos que fizemos, trabalhamos com a classe de Definição e a classe de Não-definição dividindo claramente todos os textos existentes.

Como é amplamente discutido em (Nigam, Maccallum, Thrun e Mitchell, 2000) quando temos uma classe negativa que engloba diversos tipos de texto, vamos de encontro às premissas do modelo e por isso o Aprendizado de Maquina sofre muito, levando à perda de desempenho no processo de aprendizado com textos não etiquetados. Ainda neste artigo são sugeridos dois aprimoramentos para o algoritmo com o objetivo de minimizar esta perda de desempenho: o primeiro define uma diferenciação de peso para as informações relativas às amostras diminuindo o efeito que elas têm na definição do modelo e o segundo calcula de outra forma as componentes do modelo de mistura permitindo que elas, por sua vez, sejam também compostas de uma mistura de distribuições. Desta forma relaxamos a premissa que diz que cada componente tem uma relação de um para um com as classes. Dentro de uma classe podem existir diversas subclasses e esta relação passa a ser de muitos para um.

Com o objetivo de explorar este raciocínio de uma outra forma fizemos um segundo grupo de testes utilizando um “Corpus Estendido”. Neste Corpus separamos da classe negativa um dos subtipos que antes lhe pertencia. Para nos apoiar na criação deste Corpus utilizamos a ontologia de Tipos de ALOs apresentada no primeiro capítulo visando obter com isso uma melhora no desempenho do classificador e do aprendizado semi-supervisionado.

5.3.Resultados

A seguir apresentaremos os resultados obtidos nos experimentos feitos separando-os em tópicos que abordarão cada uma das avaliações propostas.

5.3.1.Corpus Definição

No primeiro Corpus de Definição, fizemos diversos experimentos com objetivos distintos. O primeiro levantou as diferenças encontradas com a variação das *features* (quantidade de atributos) utilizadas no modelo. No segundo avaliamos especificamente o processo do aprendizado semi-supervisionado e finalmente avaliamos a utilização do Corpus desenvolvido para a classificação de arquivos com origens totalmente distintas dos textos utilizados no Corpus.

5.3.1.1.Seleção de *features* do modelo

O primeiro experimento utilizou quatro casos distintos. Dois deles utilizaram o caso mais simples, onde cada palavra é considerada separadamente, e os outros dois agruparam as palavras duas a duas o que traria maiores informações quanto à ordenação delas.

Modelagem	**Qtd**	**Accuracy**	***Recall***	***Precision***
1 Grama	7797	90,55%	96,007%	84,8866%
1 Grama com *stemming*	7110	91,42%	95,946%	86,3241%
2 Grama	17784	75,62%	86,969%	60,9113%
2 Grama com *Stemming*	17601	76,15%	87,458%	61,4078%

Tabela 3 – Resultados com o Corpus inicial avaliação de *features*

Podemos observar que quando utilizamos o recurso de *stemming* as classificações apresentam um desempenho um pouco melhor, mas podemos concluir que o uso do recurso de 2-grama diminui a performance da classificação.

A partir destes resultados resolvemos reduzir nossos experimentos as *features* que apresentaram maior desempenho. Passamos a utilizar então sempre unigrama e o algoritmo de *stemming* nos experimentos seguintes.

5.3.1.2. Aprendizado Semi-Supervisionado

Objetivando apresentar o aprendizado semi-supervisionado ocorrendo em um experimento fizemos um gráfico com o valor da Accuracy para todas as amostras de ambas as classes.

Nestes gráficos podemos observar que a Accuracy aumenta à medida que a quantidade de amostras é acrescentada aos experimentos. Cada uma das curvas dos gráficos abaixo equivale a uma quantidade diferente de exemplos utilizados no cálculo do modelo básico. Todas as curvas de aprendizado do primeiro gráfico estabilizaram-se em um patamar e a sua maior parte em torno de 90%. Podemos então observar que a partir de um número específico de exemplos, as curvas de aprendizado não são muito afetadas mantendo-se no mesmo nível ao acrescentar amostras.

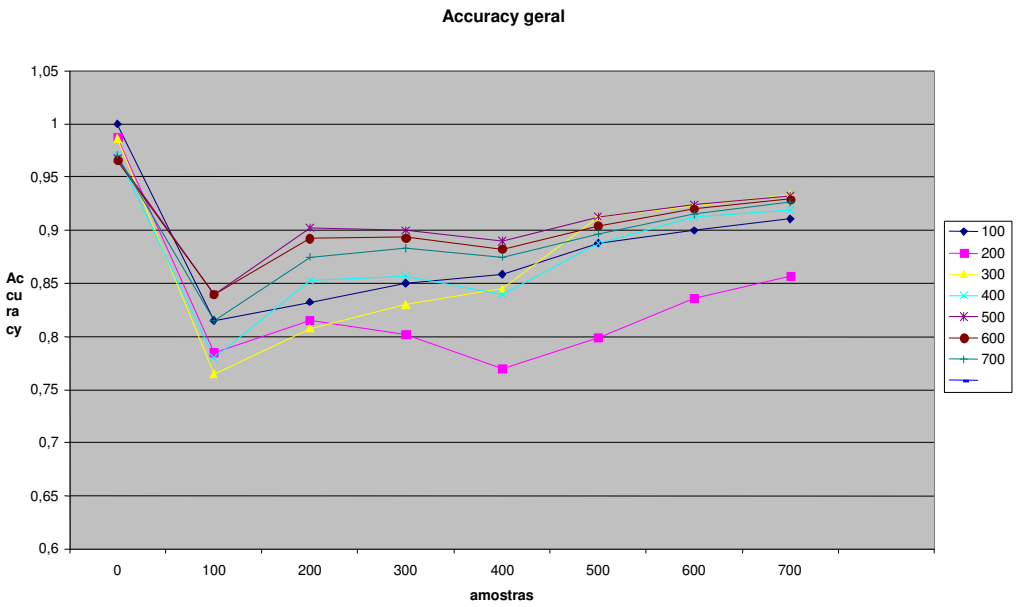

Figura 244 – Accuracy Corpus Definição

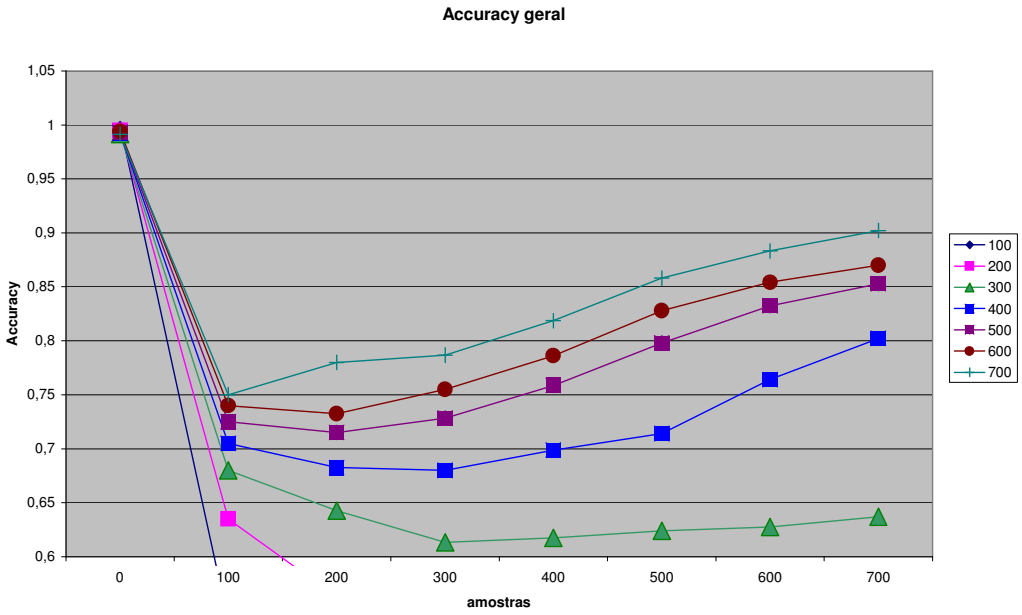

Figura 255- Accuracy Corpus Aumentado

Note que as curvas no primeiro gráfico são bem diferentes da do segundo. Cada um destes experimentos utilizou um corpus diferente e esta variação nos mostra como o aprendizado semi-supervisionado é sensível à forma como os exemplos são divididos. No Corpus Aumentado, apesar de não termos obtido nenhum ganho para a quantidade de exemplos menor que 400, tivemos uma melhora bem maior para as curvas a partir de 400 exemplos, chegando a ter uma melhora de 17%. Este ganho praticamente se estabiliza a partir de 500 exemplos sugerindo ser a melhor escolha.

Abaixo segue tabela com o cálculo dos ganhos:

Exemplos	**100**	**200**	**300**	**400**	**500**	**600**	**700**
100 amostras	54%	64%	68%	71%	73%	74%	75%
700 amostras	46%	48%	64%	80%	85%	87%	90%
Ganho	-17%	-31%	-7%	12%	15%	15%	17%

Tabela 4 – Ganhos de desempenho com o Aprendizado Semi-Supervisionado

5.3.1.3.Classificação de arquivos

Criamos então, manualmente, alguns arquivos com sentenças de um tipo específico para serem utilizados em experimentos e em coleta de métricas para

avaliação da classificação. Definimos ser necessário utilizar sentenças parecidas com as utilizadas nos exemplos e também utilizar outras sentenças originárias de textos bem diferentes. Desta forma seria possível avaliar corretamente a classificação, pois no arquivos de teste não teríamos os mesmos textos que os encontrados nos textos utilizados no treinamento. Desenvolvemos finalmente os últimos dois arquivos de testes como casos extremamente difíceis, pois se tratavam de textos próprios de livros de ensino médio contendo diversos tipos de sentenças misturados.

Arquivos de teste	Classe	*Recall*	*Precision*	F1
Definições retiradas de livros de ensino médio	Definição	93%	100%	96%
Não definições retiradas de livros de ensino médio	Não Definição	30%	100%	46%
Livros de física do ensino médio (Aula sobre medidas)	Definição	100%	08,6%	21%
	Não Definição	36%	100%	53%
Livros de física do ensino médio (Aula sobre Energia)	Definição	100%	12%	16%
	Não Definição	27%	100%	42%

Tabela 5 – Resultados com o Corpus Definição

Podemos observar por estes resultados que o valor de *Recall* se manteve sempre elevado para a classe de Definição indicando que a grande maioria das definições existentes nos textos foram corretamente indicadas pela classificação. Mas o valor da métrica *Precision* se apresentou bastante baixo em casos mais complexos indicando que diversas partes dos textos que não eram definição foram mal classificadas.

5.3.2.Corpus Estendido

O Corpus Estendido tem mais uma classe que diferencia outro subconjunto da classe negativa. Para escolher esta nova classe utilizamos a ontologia de LOs descrita em capítulos anteriores. Escolhemos a classe Lei, pois este tipo de texto era encontrado com muita freqüência e também foi, por diversas vezes, classificado como definição.

Inicialmente fizemos um Corpus com duas classes: Lei e Não-Lei e depois trabalhamos com um Corpus com três classes: Definição, Lei e Outros.

5.3.2.1.Desempenho da Classificação

Para mantermos um domínio comum entre os experimentos refizemos o Corpus de Definição mantendo os mesmos exemplos nos três casos. Utilizamos um arquivo com exemplos de Lei, três arquivos com exemplos de Definição e dois arquivos com exemplos de Não-Definição que eram textos retirados dos dois livros mencionados anteriormente. Para o Corpus de Lei, apenas o arquivo de exemplos de Lei foi classificado como Lei, os outros ficaram na classe negativa. Para o Corpus de Definição o arquivo de Lei e os livros foram incluídos na classe negativa e os três arquivos de definição ficaram na classe Definição. E finalmente no Corpus Estendido definimos três classes, o arquivo de Lei ficou na classe Lei, os três arquivos de Definição na Classe de Definição e os dois livros na classe negativa.

Para testar igualmente os três Corpus, utilizamos a mesma quantidade de exemplos, limitado à quantidade reduzida que tínhamos disponível para a classe de Lei. Neste caso utilizamos apenas 100 exemplos em todos os experimentos. A partir do modelo criado com estes 100 exemplos, classificamos 1100 amostras. Abaixo seguem os resultados:

	Corpus de Definição	Corpus de Lei	Corpus de Definição e Lei
Precision Definição	97%		87%
Precision Lei		16%	41%
Precision Negativa	73%	99%	97%
Recall Definição	46%		83%
Recall Lei		76%	72%
Recall Negativa	99%	90%	93%
F1 Definição	62%		85%
F1 Lei		26%	46%
F1 Negativa	84%	94%	95%
Accuracy	77,7273%	89,2273%	89,2332%

Tabela 6 – Comparação do Corpus Definição com o Corpus Estendido

É importante salientar que os exemplos de Lei foram utilizados nos três experimentos sendo que no primeiro eles fizeram parte dos exemplos da classe negativa e, no segundo e no terceiro, da classe positiva de Lei.

Ao classificar as 27 amostras de Lei a *precision* no Corpus Estendido ficou bem maior e o *Recall* um pouco menor. Obtivemos então uma melhora considerável na medida F1 para estas amostras. Para as amostras de definição tivemos uma queda na precisão quando utilizamos o Corpus Estendido, mas o *Recall* praticamente dobrou. Obtivemos novamente uma melhora considerável na medida F1 para estas amostras. Novamente é importante frisar que os exemplos da classe negativa foram diferentes nos três casos.

Resolvemos então repetir os experimentos utilizando sempre os mesmos exemplos para termos outra avaliação. A quantidade de 99 exemplos foi dividida da seguinte forma:

	Corpus de Definição	Corpus de Lei	Corpus de Definição e Lei
Classe Definição	33		33
Classe Lei		27	27
Classe Negativa	27 + 39	33 + 39	39

Tabela 7 – Distribuição dos exemplos nos Corpus

Utilizando as mesmas 1000 amostras nos três experimentos, obtivemos os seguintes resultados:

	Corpus de Definição	Corpus de Lei	Corpus de Definição e Lei
Precision Definição	98%		89%
Precision Lei		75%	64%
Precision Negativa	70%	98%	95%
Recall Definição	19%		89%
Recall Lei		22%	67%
Recall Negativa	99%	99%	95%
F1 Definição	32%		89%
F1 Lei		34%	65%
F1 Negativa	82%	99%	95%
Accuracy	72%	98%	93%

Tabela 8 – Comparação do Corpus Definição com o Corpus Estendido

Novamente podemos observar que o valor de *Recall* tanto para a classe de definição quanto para a classe de Lei aumentou no Corpus Estendido. A precisão caiu em ambos os casos, mas a medida de F1 aumentou.

Podemos concluir que o Corpus Estendido parece apresentar maior desempenho no reconhecimento das classes. A variação nos resultados ocorreu devido à variação nos exemplos que foram utilizados e como para a classe de lei não tínhamos um Corpus muito extenso para trabalhar tivemos que reduzir a opção de testes e consequentemente de resultados, o que não é o ideal neste tipo de experimento. Mas a tendência a melhora de desempenho para o Corpus Estendido parece bastante clara.

5.3.2.2. Desempenho do Aprendizado Semi-Supervisionado

Objetivando observar o que acontece com o desempenho do aprendizado semi-supervisionado quando utilizamos o Corpus Estendido fizemos um gráfico com o valor da Accuracy para todas as amostras de ambas as classes em cada um dos três Corpus utilizados anteriormente. Os experimentos utilizaram os mesmos 100 exemplos utilizados no experimento anterior na mesma distribuição entre as classes. Variamos então a quantidade de amostras acrescidas ao modelo e obtivemos os resultados a seguir:

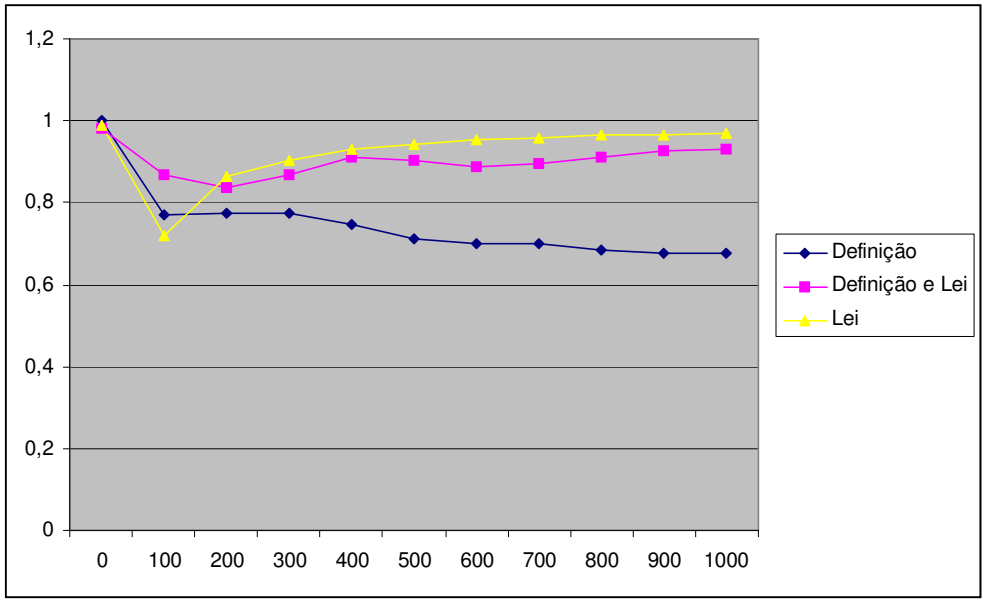

Figura 266 - Comparação da curva de Aprendizado - Valores de Accuracy

Mais uma vez podemos comprovar que, quanto mais as premissas usadas pelo modelo Bayesiano de misturas forem verídicas para o experimento, melhor é a curva de aprendizado. Quando utilizamos o Corpus de Definição perdemos desempenho do classificador com o uso de amostras. Neste Corpus a classe negativa possuía textos de Lei e Outros misturados. Quando utilizamos o Corpus de Lei o aprendizado também foi ruim. E finalmente para o Corpus Estendido a curva de aprendizado foi bastante boa. A seguir apresentamos os gráficos.

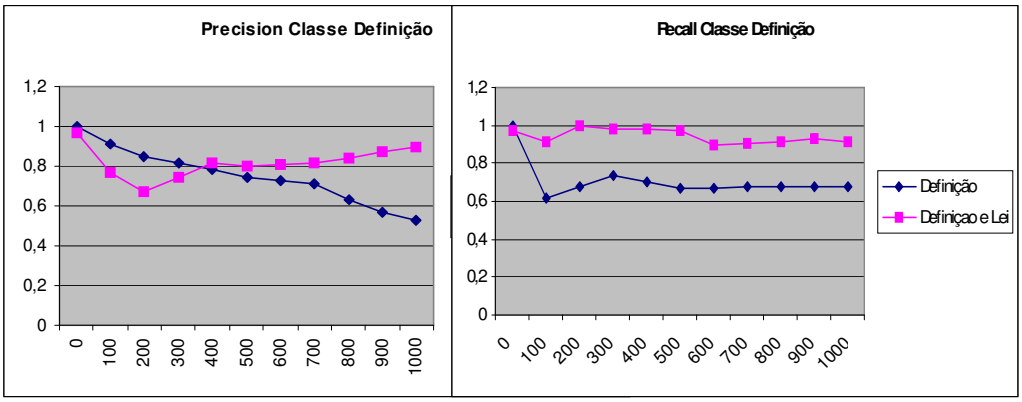

Figura 277 – *Precision* e *Recall* para Classe Definição

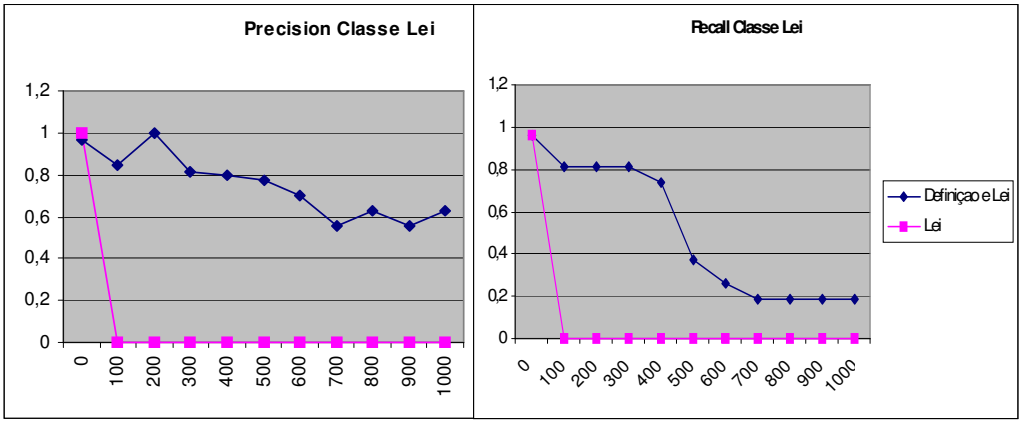

Figura 288– *Precision* e *Recall* para Classe Lei

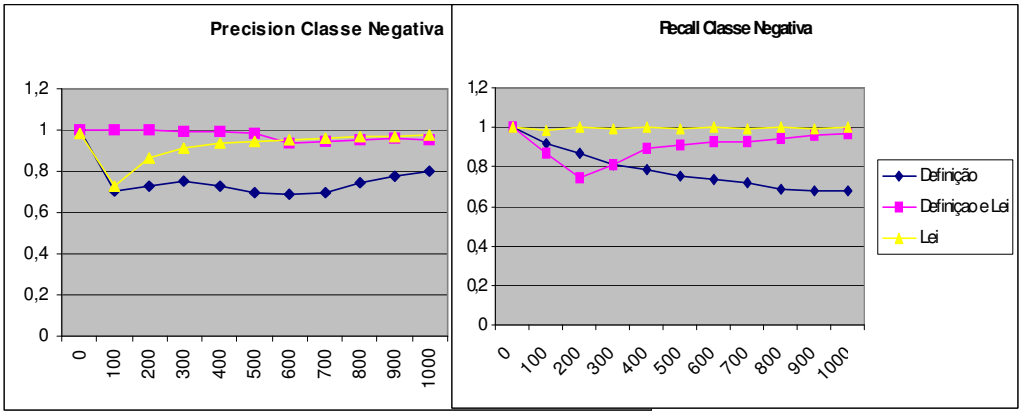

Figura 299 – *Precision* e *Recall* para Classe Negativa

6. Trabalhos Relacionados

Durante a pesquisa desenvolvida para a confecção deste trabalho encontramos uma vasta literatura relativa a ALOs. Muitos trabalhos já foram desenvolvidos utilizando a abordagem de armazenamento de LOs (Pereira et. al., 2003), (Melo, et al., 2005), (Silvia, et al., 2005), (Leal et al., 2006), (Gomes et al., 2006), (Baruque et. Al., 2006). Mas nesta estrutura que se criou para a manipulação e armazenamento destes objetos, existe uma lacuna relativa ao processo de geração de conteúdo a ser utilizado na criação dos LOs. O único trabalho encontrado que abordou este problema foi o (Gomes et al., 2006).

Em paralelo pesquisamos diversos projetos que objetivavam a especulação em torno da melhor técnica de aprendizado de máquina aplicada na classificação de textos. Muitos dos trabalhos que estudamos visavam à comparação entre diversos algoritmos distintos utilizados em um mesmo problema. Outros visavam especificamente à criação ou o estudo aprofundado de um processo de classificação, utilizando para isso algoritmos de classificação específicos e propondo aprimoramentos.

Os problemas tratados nestes trabalhos foram os mais diversos, cada um envolvendo um domínio diferente. Cada um almejava reconhecer um tipo diferente de classificação dos textos e por isso utilizava um Corpus específico ao domínio em questão.

Alguns trabalhos focalizaram as pesquisas na comparação de resultados obtidos quando variavam as *features* utilizadas pelos algoritmos na criação dos modelos. Nestes casos, opções de parametrização eram oferecidas no pré-processamento que era feito nos textos ao transformá-los em tabelas atributo-valor a ser lida pelos algoritmos. Um exemplo de opções de parametrização é a utilização de n-gramas ou a utilização de recursos como *stemming*.

Outros trabalhos preferiram investigar o processo de aprendizado ocorrido no algoritmo à medida que novas amostras foram sendo acrescentadas ao modelo, permitindo concluir a utilidade do aprendizado semi-supervisionado. Nestes trabalhos variou-se também a quantidade de exemplos utilizados no Corpus inicial permitindo avaliar o tamanho ideal necessário do Corpus de exemplos para se atingir o melhor desempenho neste tipo de aprendizado.

A seguir descreveremos sucintamente cada um dos trabalhos escolhidos e faremos algumas comparações levando em consideração cada um destes aspectos descritos anteriormente.

6.1. Descrição dos trabalhos

O primeiro trabalho que descreveremos foi o propulsor do presente (Gomes et al., 2006). É uma tese de doutorado desenvolvida no nosso grupo de pesquisa que teve como motivação principal pesquisar, estudar, comparar e propor um modelo, uma arquitetura, um ambiente de software, capaz de integrar dados de Bibliotecas Digitais - DLs e de Sistemas de Aprendizagem. Utilizou-se para isso várias tecnologias, tal como: mediadores para fazer integração, mineração de texto para extração do conteúdo dos documentos da biblioteca digital e uma ontologia para a integração semântica dos mesmos.

O processo proposto neste trabalho para a mineração de texto e extração de conteúdo enfocava a utilização de uma metodologia de raciocínio dedutivo. Foram definidas regras para extrair automaticamente as definições contidas nos documentos digitais, com base em regras genéricas. As regras geralmente foram definidas através de uma representação comum na forma situação/ação. Um exemplo deste tipo de representação são regras do tipo “Se [condição] Então [ação]”.

Os resultados alcançados neste trabalho são descritos na tabela a seguir:

No	Referência	Def. Reais	Definições extraídas	Lixo	Definições Corretas	Precisão	*Recall*	F1
1	(Portugal, 2004)	5	16	11	4	0,25	0,8	0,38
2	(Ochi, 2004)	8	16	12	4	0,25	0,50	0,33
3	(Silva, 2004a)	6	2	1	1	0,5	0,16	0,25
4	(Lopes, 2004)	0	0	0	0	0	0	0
5	(Silva, 2004b)	9	5	0	5	1	0,55	0,71
6	(Penha, 2004)	8	13	8	5	0,38	0,62	0,47
7	(Wedemann,2004)	1	3	1	2	0,66	2	0,71
8	(Barbosa, 2004)	10	2	1	1	0	0,1	0
					Média	0,38	0,59	0,35

Tabela 9 – Resultados do primeiro trabalho relacionado

O segundo trabalho analisado (Matsubara e Monard, 2006) fez parte de um projeto desenvolvido em um grupo de pesquisa de Inteligência Computacional do ICMC-USP que teve como objeto de pesquisa principal um algoritmo de aprendizado semi-supervisionado chamado de CO-TRAINING, proposto por (Blum e Mitchell, 1998) com o objetivo de utilizá-los em problemas de classificação de texto. Adicionalmente foi desenvolvido um ambiente computacional para o pré-processamento de textos, denominado PréText.

O foco principal deste trabalho foi a analise de resultados obtidos utilizando-se o aprendizado semi-supervisionado que introduz a idéia de utilizar um pequeno número de exemplos rotulados, já que é geralmente caro e difícil de obter, e um grande número de exemplos não-rotulados, os quais se encontram facilmente disponíveis, com o objetivo de rotular mais destes exemplos para melhorar o desempenho de algoritmos de Aprendizado de Máquina.Assim como este, existem outros trabalhos que tiveram como objetivo principal a análise do aprendizado semi-supervisionado, um exemplo é o artigo que já referenciamos diversas vezes (Nigam, Mccallum, Thrun e Mitchell, 2000).

O CO-TRAINING desenvolveu três bases de textos que foram utilizadas nos experimentos. O objetivo era a classificação dos textos em duas classes relativas aos temas abordados nos textos e o último Corpus dividiu os textos em duas classes: course e non-course separando os textos que eram referentes a cursos oferecidos em universidades. Para a avaliação do processo de aprendizado semi-supervisionado podemos destacar os seguintes resultados:

	% de exemplos iniciais	Erro médio	Accuracy	Dif
Primeira iteração	1 %	20.1	79.9	
Última iteração		1.6	98.4	23
Primeira iteração	2 %	12.4	87.6	
Última iteração		1.4	98.6	12,5
Primeira iteração	5 %	6	94	
Última iteração		1.4	98.6	4,9
Primeira iteração	7 %	2.7	97.3	
Última iteração		1.4	98.6	1,3
Primeira iteração	10 %	4.7	95.3	
Última iteração		1.1	98.9	3,8

Tabela 10 – Resultados do segundo trabalho relacionado

Um terceiro trabalho que gostaríamos de citar foi desenvolvido na PUC-Rio (Steinbruch, 2006). Ele propõe dois algoritmos de classificação automática de textos baseados no algoritmo Multinomial Naive Bayes e sua aplicação em um ambiente on-line de classificação automática de notícias com realimentação de relevância pelo usuário, combinando técnicas de aprendizado de máquina e mineração de textos.

Particularmente este trabalho foca, diferente da maioria das abordagens propostas, não apenas a resolução de um problema de classificação binária de textos, mas sim a consideração do caso mais geral onde existe a possibilidade de um documento estar associado a mais de um rótulo.

Para testar a eficiência dos algoritmos propostos, foram realizados testes na base de dados da Reuters composta de um conjunto de 21.578 notícias que foram publicadas na rede de notícias Reuters, em 1987, e classificadas de acordo com 135 categorias, a maioria sobre economia e negócios. Adicionalmente utilizou-se a base de dados Ohsumed que é um subconjunto de 348.566 documentos da base MEDLINE (uma base on-line de textos sobre medicina), compilada por William Hersh e proveniente de 270 jornais médicos por um período de cinco anos (1987- 1991).

Como o domínio e o Corpus utilizados foram muito diferentes dos nossos fica complicado comparar os valores das métricas alcançados. Podemos

destacar como resultados interessantes para comparação os parâmetros e quantidade de atributos que alcançaram melhores resultados.

Utilizando a base Reuters(10) onde são considerados apenas as 10 classes que possuem a maior quantidade de exemplos, chegou-se aos resultados que se segue utilizando quase 7000 exemplos para o treinamento do modelo multinomial.

Micro *Recall*	94,26
Micro *Precision*	92,11
Micro F1	93,17
Macro *Recall*	88,98
Macro *Precision*	83,90
Macro F1	85,86

Tabela 11 - Resultados do terceiro trabalho relacionado

Ainda neste trabalho são listadas como sugestões para trabalhos futuros dois tópicos que foram abordados aqui:(1) “Possibilitar que o aprendizado seja semi-supervisionado, reduzindo a necessidade de o usuário associar para cada documento um conjunto de categorias, tarefa que pode ser bastante trabalhosa e suscetível a erros.” (2) “Inclusão de relações entre categorias, ou seja, permitir que o usuário apresente ao sistema relações entre categorias e o sistema agregue tais informações ao aprendizado.

Utilizando o algoritmo EM acrescentamos o aprendizado semi-supervisionado ao modelo multinomial reduzindo a tarefa de etiquetagem dos exemplos. Sobre a segunda sugestão sugerimos aqui a utilização de uma ontologia para a representação das relações entre as classes. Procuramos também mostrar que o uso dela com apoio a definição das classes do modelo favoreceu o processo de aprendizado de máquina. Ainda nesta linha de pesquisa outros trabalhos futuros serão sugeridos mais adiante.

6.2.Comparação com outros trabalhos

A tarefa de obter LOs a partir de textos diversos é um problema razoavelmente antigo, mas ainda não muito explorado. Diversos trabalhos relacionados ao tratamento de LOs podem ser encontrados, mas todos se baseiam em bancos de dados de LOs já desenvolvidos, não focando no

processo de extração de textos para a criação destes LOs, mas nos processos de geração manual, reutilização, troca, recuperação e armazenamentos deles.

O domínio do problema que estamos analisando trata de uma área extremamente abstrata, onde o foco principal é transmissão de um conceito. Queremos encontrar textos que transmitam conhecimento, que é algo bastante complexo de se descrever. Na verdade não estamos à procura de um tema como a maioria dos trabalhos na área de classificação de textos, mas sim de uma forma de texto, de um propósito. Estamos querendo capturar o objetivo instrucional de cada parte de um texto que pertença a um conteúdo didático qualquer. Por este motivo, as classes que propusemos para a classificação do texto tinham como atributos diferenciadores características muito difíceis de serem capturadas.

Sob este aspecto o único trabalho que encontramos com um propósito parecido com o nosso foi o primeiro trabalho abordado no tópico anterior e que foi o empregado no primeiro Corpus que desenvolvemos. Conseguimos alcançar um desempenho maior, no máximo 75% de *recall* contra 100% do presente. Os valores máximos de 66% de *precision* contra 8% do nosso, por outro lado pode querer condenar o nosso processo, mas no caso de classificação binária a medida mais importante a ser analisada é o *recall*, pois ela indica mais claramente o desempenho obtido quando a classificação se aproxima mais de uma filtragem que é o caso da seleção de LOs de definição.

Refizemos também um dos experimento de (Gomes et al., 2006) onde são extraídas de um artigo todas as definições existentes. Para isso utilizamos o Corpus de Definição desenvolvido no nosso trabalho. Este Corpus não utilizou nenhum exemplo parecido com os textos encontrados neste artigo, mas apesar disso, para o texto (Ochi, 2004) encontramos um *Recall* de 61%, ou seja, 61% das definições que foram encontradas numa análise manual foram selecionadas pelo algoritmo. No trabalho (Gomes et al., 2006) foi indicado um *Recall* de 50%. A precisão alcançou 37% e a medida F1 ficou em 46% ambas as medidas também maiores.

O algoritmo CO-TRAINING utilizado em (Matsubara, Monard, 2006) apresenta um método de escolha dos exemplos a serem rotulados diferentes dos métodos utilizados pelo algoritmo descrito neste trabalho, mas como análise

do processo de aprendizado semi-supervisionado podemos observar nos nossos resultados a mesma tendência.

% de exemplos iniciais	Accuracy CO-TRAINING	% de Aumento na Accuracy CO-TRAINING	Accuracy SQLLOMining	% de Aumento na Accuracy SQLLOMining
1%	79.9 98.4	19	50 34.1	-32
2%	87.6 98.6	11	50.1 34.2	-32
5%	94 98.6	5	50.2 34.4	-34
7%	97.3 98.6	1	78.8 81.6	4
10%	95.3 98.9	4	88.1 92.7	5
20%			92.3 95.1	3
30%			93.9 96.1	2
40%			93.6 95.4	2

Tabela 12 – Comparação de resultados com segundo trabalho relacionado

Ambos os algoritmos aumentam a Accuracy com a adição de amostras ao modelo inicial gerado a partir dos exemplos. À medida que a quantidade de exemplos iniciais cresce o ganho no aprendizado com as amostras diminui e sempre podemos encontrar uma faixa onde o aprendizado é alto e a quantidade de exemplos ainda se mantém pequena sugerindo ser a melhor escolha.

Podemos também observar que, diferente dos trabalhos mencionados (Matsubara, Monard, 2006) e (Nigam, Mccallum, Thrun e Mitchell, 2000), nos nossos experimentos tivemos que utilizar uma quantidade de exemplos iniciais bem maior para obter ganho no aprendizado com as amostras. Podemos destacar os resultados apresentados na tabela 4 que indicam a necessidade de pelo menos 400 exemplos para obter 12% de ganho com a adição de amostras. Para o trabalho (Nigam, Mccallum, Thrun e Mitchell, 2000) utilizando apenas 40

exemplos o aprendizado ocorre com um ótimo ganho de 60% e na tabela acima podemos ver que para pequenas quantidades de exemplos, os ganhos já são bem significativos também para o trabalho de (Matsubara, Monard, 2006).

Podemos atribuir esta diferença nos resultados principalmente ao fato do domínio do problema ser bem diferente nos três casos. Tanto que podemos encontrar em (Matsubara, Monard, 2006) resultados bem ruins para o experimento mais parecido com o nosso, onde foram classificados em course e non-course textos que eram referentes a cursos oferecidos em universidades. Neste experimento, além da subjetividade na definição das classes o fato da classe course possuir muito mais casos dentre os exemplos, atrapalha demais o processo de aprendizado com este tipo de algoritmo.

Conforme também mencionado em (Steinbruch, 2006), o modelo de misturas é muito sensível ao fato de não termos a quantidade de exemplos balanceada entre as classes. Nos experimentos feitos com o Corpus de Lei e Não-Lei tivemos claramente este mesmo problema, pois dispúnhamos de poucos exemplos. Pudemos observar uma diminuição de valores dos resultados quando a quantidade de exemplos ultrapassou o limite de 100. Nestes casos a quantidade de exemplos de Lei não chegava a 30% do total. Entre os 100 exemplos utilizados apenas 27 eram de Lei. À medida que acrescentamos exemplos os valores foram diminuindo o que não aconteceu com o Corpus de Definição que possuía 342 exemplos. Para o Corpus de Definição o limiar de 30% só foi alcançado quando utilizamos 1000 exemplos.

Exemplos	36	50	100	200	300
Accuracy	79%	81%	95%	88%	80%
Recall - Classe Lei	91%	93%	80%	55%	25%
Precision - Classe Lei	63%	63%	100%	100%	93%
F1 - Classe Lei	72%	74%	88%	71%	41%

Tabela 13 – Resultados obtidos com o Corpus Lei variando a quantidade de exemplos

7. Conclusão

A motivação principal deste trabalho é a busca de um processo que permita que um usuário pesquise em documentos, LOs que ele possa reutilizar na composição de seu material didático. Partindo do tipo de LO desejado, que especificaria a sua proposta instrucional, e de palavras-chave indicando o tema de interesse, o sistema proposto encontrará partes elementares de documentos que contenham o assunto procurado no formato especificado (por exemplo, uma definição). Deste modo facilitamos a criação de ALOs que poderão ser recuperados em consultas integradas durante a composição de LOs complexos (aulas, cursos etc...).

Propusemos a aplicação de um modelo onde o objeto LO é segmentado em pedaços menores, permitindo facilmente a reutilização desta pequena porção de conhecimento, agora apreendida em um ALO, em diversas situações. Podemos, por exemplo, utilizar a definição de um termo ou palavra, em qualquer conteúdo instrucional que esteja tratando de algum conceito relacionado a este termo.

Como apoio na definição, com maior nível semântico, do que pode ser considerado um ALO propusemos o uso de uma Ontologia de Objetos de Aprendizagem. Utilizamos técnicas de aprendizado de máquina para apreender dos exemplos, informações sobre os textos que queremos encontrar, gerando um Corpus e um Modelo Bayesiano de Misturas Multinomiais.

Utilizando a solução proposta fizemos alguns experimentos e alcançamos um nível de precisão nesta classificação dentro do esperado, com valores de *recall* altos para as classes positivas indicando que o algoritmo encontrou a grande maioria dos textos que deveriam ser encontrados. Por outro lado, diversos resultados dos experimentos principalmente relativos a métrica *precision* nos mostraram as dificuldades que podem ser encontradas na aplicação do aprendizado de máquina a este problema.

7.1.Contribuições

A principal contribuição deste trabalho foi o desenvolvimento de um processo para obtenção de LOs que foi formalizado com a criação do sistema SQLLOMining. Este sistema propiciou um ambiente para a execução de experimentos que poderá ser utilizado como um laboratório, dando prosseguimento a execução de novos testes que ficam aqui listados como trabalhos futuros.

No sistema SQLLOMining foi implementado o algoritmo Expectation-Maximization (EM) associado ao modelo Bayesiano de misturas multinomiais, proposto por (Nigam, Maccallum, Thrun e Mitchell, 2000) disponibilizando um classificador de textos que executa o aprendizado semi-supervisionado, recurso bastante poderoso no processo de mineração de texto. A aplicação desta técnica ao problema de obtenção de LOs trouxe contribuições tanto para a área de aprendizado de máquina quanto para a área de e-learning.

Adicionalmente, com a realização de um experimento bastante simples utilizando o sistema SQLLOMining, pudemos observar que o uso da ontologia apoiando a criação das classes favoreceu o processo de aprendizado. A ontologia aumentou o nível semântico do modelo de LOs e proporcionou um maior grau de comprometimento em manter coerentes os tipos de ALOs utilizados na classificação. Desta forma procuramos satisfazer minimamente as premissas utilizadas na especificação do modelo de misturas permitindo a utilidade do aprendizado semi-supervisionado. Este experimento mostrou que o processo de separação dos textos existentes entre as classes possíveis é uma parte do problema de extrema importância, que nos remete ao já bastante estudado “Problema de Clusterização” (Ochi, 2004).

7.2.Trabalhos Futuros

Como trabalhos futuros existem diversos experimentos que podem trazer resultados muito interessantes sobre como a ontologia pode auxiliar no processo de obtenção de LOs. Um exemplo é analisar o que ocorreria com as métricas de desempenho se passássemos a utilizar os níveis mais altos da ontologia, como classes para o algoritmo de classificação. Intuitivamente podemos imaginar que quando trabalhamos com uma classe de um nível mais elevado perdemos

características específicas que facilitariam a sua identificação. É razoável supor que quanto mais elevado este nível for mais difícil fica a classificação.

A justificativa teórica que podemos encontrar para esta intuição está nas duas premissas relativas à geração de documentos: (1) existe um modelo de misturas, (2) existe uma correspondência de um para um entre as componentes da mistura e as classes. Para nos mantermos coerentes com estas premissas, é necessário subdividir as classes, permitindo que a relação de um para um seja verdadeira. Se utilizarmos, por exemplo, apenas uma superclasse chamada conceito englobando as suas subclasses: Definição, Lei, Fato, Procedimento etc..., e criarmos uma segunda classe chamada de Outros que englobaria todo o resto, claramente teremos uma relação de um para n entre as componentes e as classes. Neste caso esperamos obter um desempenho pior.

Outros trabalhos futuros que podem se feitos utilizando o sistema SQLLOMining:

- Aumento do corpus:
  - Permitir a execução de um n-fold cross validation aumentando a qualidade dos experimentos
  - Aumentar a quantidade de classes
- Aprimoramentos no algoritmo EM para melhorar o desempenho do aprendizado semi-supervisionado:
  - Fator de peso para as amostras
  - Mistura para cada componente - similar a utilização de mais classes indicadas pela ontologia
- Aprimoramento no pré-processamento do texto:
  - Separação dos termos mais importantes
  - Separação de n-gramas considerando todos os grupos possíveis

Finalmente podemos concluir que este trabalho também contribui para a seqüência de pesquisas voltadas para e-Learning do Laboratório TecBD do Departamento de Informática visando o descobrimento, obtenção, integração, armazenamento e compartilhamento de Learning Objects.

Referências bibliográficas

BARUQUE, L. B. ; MELO, R. N. . **Reference Model for e-Learning Governance.** In: 22nd ICDE (International Concil for Open and Distance Education), 2006. Proceedings 22nd ICDE 2006, 2006.

BARUQUE, C. B; MELO, R. N.. **Desenvolvimento de Bibliotecas Digitais de Learning Objects Utilizando Técnicas de Data Warehousing e Data Mining**, Tese de Doutorado, PUC-Rio, 2005.

BLUM, A, MITCHELL T. - **Combining labeled and unlabeled data with co-training**. ACM Press, New York, NY, 1998.

BORMAN, S. - **The Expectation Maximization Algorithm - A short tutorial**. October 2006. http://www.seanborman.com/publications/EM_algorithm.pdf - Acesso em Julho de 2007

CHAKRABARTI, S. - **Mining the Web: Discovering Knowledge from Hypertext Data**, Morgan Kaufmann, 2002

CISCO (1999) - **Cisco Systems Reusable Information Object Strategy - Version 3.0** - http://www.cisco.com/warp/public/779/ibs/solutions/learning/whitepapers/el_cisco_rio.pdf - Acesso em Julho de 2007.

DEMPSTER, P., LAIRD, N. M., and RUBIN, D. B.- **Maximum likelihood from incomplete data via the em algorithm**. Journal of the Royal Statistical Society: Series B, 39(1):1–38, November 1977.

GENNARI, J. H., MUSEN, M. A., FERGERSON, W., GROSSO, W. E., CRUBÉZY, M., ERIKSSON, H., NOY, N. F., and TU, S. W. -**The evolution of Protégé: an environment for knowledge-based systems development.** International Journal of Human-Computer Studies, 58(1):89–123, 2003.

GLEISER, M. - **A dança do universo: dos mitos de criação ao Big-Bang** – São Paulo: Companhia das Letras, 2006

GOMES, G. R. R., MELO R. N., SIQUEIRA, S. W. M., BRAZ, M. H. L. B. . **Integrated Searches over Digital Libraries and E-learning Systems.** In: WCCSETE 2006 Congresso Mundial de Educação em Engenharia, Tecnologia e Ciência da Computação, 2006, Itanhaém, Santos, 2006.

HAWKING, S. - **Uma Breve História do Tempo – Do Big-Bang aos Buracos Negros**. Editora Rocco, 1988.

J2EEBrasil - **O site da comunidade J2EE no Brasil** - www.j2eebrasil.com.br- Acesso em Julho de 2007.

LEAL, S, MELO R. N**.- Uma arquitetura para Integração de Repositórios de Objetos de Aprendizagem baseada em Mediadores e Serviços Web**. Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, 2006.

LEARNING TECHNOLOGY STANDARDS COMMITTEE **– Draft standard for learning object metadata**. Technical Report IEEE P1484.12.1/D6.4, IEEE March 2002.

LOVINS, J. B. - **Development of a stemming algorithm**. Mechanical Translation and Computacional Linguistics 11 (1–2), 22–31, 1968.

LUBELL, K. (May 2006) – **SQL Stemming algorithm**. http://www.tartarus.org/martin/PorterStemmer/tsql.txt - Acesso em Julho de 2007

MATSUBARA, E. T.; MONARD, M. C. **- O algoritmo de aprendizado semi-supervisionado co-training e sua aplicação na rotulação de documentos.** Dissertação (Mestrado em Informática) - Universidade de São Paulo, USP, Brasil, 2004.

MCCALLUM, A., E NIGAM, K. – **A comparison of event models for naive bayes text classification.** In AAAI-98 Workshop on Learning for Text Categorization. Tech. rep. WS-98-05, AAAI Press. 1998.

MCLACHLAN, G. J., & KRISHNAN, T. -**The EM Algorithm and Extensions**. John Wiley and Sons, New York 1997.

MELO, R. N. ; BARUQUE, L.B.: **A Database Approach to Partnership In Global Learning.** Proc. I PGL Database Research Conference.(2003), Disponível em: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-70/paper5.pdf

MELO, R. N. ; BARUQUE, C. B. ; BARUQUE, L. B.. **Applying Governance in e-learning: a risk-based approach.** In: IADIS - INTERNATIONAL ASSOCIATION FOR DEVELOPMENT OF THE INFOMATION, 2005, Portugal 2005.

MITCHELL, T. M. - **Machine Learning**. Boston, USA: McGraw-Hill, 1997.

NIGAM, K., MCCALLUM, A.; THRUN, S.; MITCHELL, T. **- Text classification from labeled and unlabeled documents using EM**. Machine Learning, 39(2/3): 103–134, 2000.

OCHI, L.S., DIAS, C.R., SOARES, S.S.F. – **Clusterização em Mineração de Dados.** Disponível em: http://bibliotecadigital.sbc.org.br/?module=Public&action=PublicationObjects&Subjects=154&publicationobjectid=9. Acesso em Julho de 2007.

PAPOULIS, A. **Probability, Random Variables, and Stochastic Processes.** 2nd ed. New York: McGraw-Hill, 1984.

PEREIRA, L. A. M., PORTO, F. A. M., MELO, R. N. **Objetos de Aprendizado Reutilizáveis (RLOs):** conceitos, padronização, uso e armazenamento. Monografia em Ciência da Computação N° 10/03, Departamento de Informática, PUC-Rio, 2003.

PORTER, M. - **An algorithm for suffixing stripping**. Program 14(3), 130–137, 1980.

SILVA, D. S.; MELO, R. N.; SIQUEIRA, S. W. M.; BRAZ, M. H. L. B. **Uma Linguagem para Especificação de Seqüências de Objetos de Aprendizagem.** In: 3ª CONFERÊNCIA DO PGL CONSOLIDANDO EXPERIÊNCIAS EM E.LEARNING, 2005, São Paulo, Proceedings, 2005.

SOUZA, W. B. - **Tutorial - Struts Framework** - http://www.j2eebrasil.com.br/jsp/artigos/artigo.jsp?idArtigo=0011

SPARK-JONES, K., WILLET, P.- **Readings in Information Retrieval**. San Francisco:Morgan Kaufmann, 1997.

STEINBRUCH, D., SCHWABE D., MILIDIU R.- **Um estudo de algoritmos para classificação automática multirótulo de textos utilizando Naive-Bayes**, Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, 2006

ULLRICH, C. - **The learning-resource-type is dead, long live the learning-resourcetype!** - Learning Objects and Learning Designs, 1(1):7-15.2005. Disponível em: http://www.ags.uni-sb.de/~cullrich/publications/Ullrich-LearningResource-LOLD-2005.pdf. Acesso em Julho de 2007.

ULLRICH C. - **Description of an instructional ontology and its application in web services for education**. In Proceedings of Workshop on Applications of Semantic Web Technologies for E-learning, SW-EL'04, pages 17-23, Hiroshima, Japan, 2004. Disponível em: http://www.ags.uni-sb.de/~cullrich/publications/Ullrich-InstructionalOntology-SWEL-2004.pdf. Acesso em Julho de 2007.

VERBERT, K. and DUVAL, E. **-Towards a Global Component Architecture for Learning Objects: A Comparative Analysis of Learning Object Content Models**.2002. Disponível em: http://www.cs.kuleuven.ac.be/~hmdb/publications/files/pdfversion/41315.pdf. Acesso em Julho de 2007.

WILEY, D. A. - **Learning Object Design And Sequencing Theory**. Unpublished doctoral dissertation, Brigham Young University, 2000. Disponível em: http://opencontent.org//docs/dissertation.pdf. Acesso em Julho de 2007.

W3C - **World Wide Web Consortium**, http://www.w3.org/. Acesso em Julho de 2007.

Strutus - http://struts.apache.org/api/index.html. Acesso em Julho de 2007.

Apêndice I

Funções para Algoritmo Porter


```
/****** Object: Table [dbo].[tblPorterStemming] ******/
CREATE TABLE [dbo].[tblPorterStemming]
( [Step] [int] NOT NULL ,
[Ordering] [int] NOT NULL ,
[phrase1] [nvarchar] (15) NOT NULL ,
[phrase2] [nvarchar] (15) NULL
) ON [PRIMARY]
```

```
GO


CREATE FUNCTION [dbo].[fnPorterCVCpattern]
( @Word nvarchar(4000) )
RETURNS nvarchar(4000)
AS
BEGIN
--local variables
      DECLARE @Ret nvarchar(4000), @i int

--checking each character to see if it is a consonent or a vowel.
also inputs the information in const_vowel
      SELECT @i = 1, @Ret = ''''

      WHILE @i <= LEN(@Word)
      BEGIN
            IF CHARINDEX(SUBSTRING(@Word,@i,1), ''aeiou'') > 0
            BEGIN
                  SELECT @Ret = @Ret + ''v''
            END

-- if y is not the first character, only then check the previous
character
            ELSE IF SUBSTRING(@Word,@i,1) = ''y'' AND @i > 1
            BEGIN

--check to see if previous character is a consonent
                  IF CHARINDEX(SUBSTRING(@Word,@i-1,1), ''aeiou'')
                  = 0
                        SELECT @Ret = @Ret + ''v''
                  ELSE
                        SELECT @Ret = @Ret + ''c''
            END
            Else
            BEGIN
                  SELECT @Ret = @Ret + ''c''
            END
            SELECT @i = @i + 1
      END

      RETURN @Ret
END
'
END


GO
CREATE FUNCTION [dbo].[fnPorterStep]
( @step int, @InWord nvarchar(4000) )
RETURNS nvarchar(4000)
AS

BEGIN
      DECLARE @Ret nvarchar(255)
      DECLARE @Phrase1 NVARCHAR(15), @Phrase2 NVARCHAR(15)
      DECLARE @CursorName CURSOR

-- DO some initial cleanup
      SELECT @Ret = @InWord
```

```
-- Create Cursor for Porter Step
      SET @CursorName = CURSOR FOR
      SELECT phrase1, phrase2
      FROM tblPorterStemming
      WHERE Step = @step
      AND RIGHT(@Ret,LEN(Phrase1)) = Phrase1
      ORDER BY Ordering

      OPEN @CursorName
-- Do Step 1

      FETCH NEXT FROM @CursorName INTO @Phrase1, @Phrase2

      WHILE @@FETCH_STATUS = 0
      BEGIN
            --
            IF RIGHT(@Ret ,LEN(@Phrase1)) = @Phrase1
            BEGIN
                  SELECT @Ret = LEFT(@Ret, LEN(@Ret) -
                  LEN(@Phrase1)) + @Phrase2
                  BREAK
            END

            FETCH NEXT FROM @CursorName INTO @Phrase1, @Phrase2
      END

-- Free Resources

      CLOSE @CursorName

      DEALLOCATE @CursorName

      return @Ret

END
'
END

GO


CREATE FUNCTION [dbo].[fnPorterCountPosV]
( @Word nvarchar(4000) )
RETURNS tinyint
AS

BEGIN
--A \consonant\ in a word is a letter other than A, E, I, O or U,
and other

--declaring local variables
      DECLARE @pattern nvarchar(4000), @ret tinyint, @i int, @flag
      tinyint

--initializing
      SELECT @ret = 1, @flag = 0, @i = 1

      If Len(@Word) > 0
      BEGIN
--find out the PosV
            SELECT @pattern = dbo.fnPorterCVCpattern(@Word)
```

```
--se a palavra comeca com duas ou mais vogais, posV esta na
primeira consoante;
            IF SUBSTRING(@pattern,1,2) = ''vv''
            BEGIN
                  set @i = 3

                  WHILE @i <= LEN(@pattern)
                  BEGIN
                        IF SUBSTRING(@pattern,@i,1) = ''c''
                        BEGIN
                              IF (SUBSTRING(@pattern,@i-1,1) =
                              ''v'')
                                    BREAK
                        END
                        SELECT @ret = @ret + 1
                        SELECT @i = @i + 1
                  END
            END

--se comeca com vogal-consoante, posV esta na segunda vogal;
            IF SUBSTRING(@pattern,1,2) = ''vc''
            BEGIN
                  set @i = 3

                  WHILE @i <= LEN(@pattern)
                  BEGIN
                        IF SUBSTRING(@pattern,@i,1) = ''v''
                              BREAK
                        SELECT @ret = @ret + 1
                        SELECT @i = @i + 1
                  END
            END

--se comeca com consoante, posV esta na primeira vogal ou na
terceira letra que esta mais a direita.

            IF SUBSTRING(@pattern,1,1) = ''c''
            BEGIN
                  set @i = 1

                  WHILE @i <= 3
                  BEGIN
                        IF SUBSTRING(@pattern,@i,1) = ''v''
                              BREAK
                        SELECT @ret = @ret + 1
                        SELECT @i = @i + 1
                  END
            END
      END
      RETURN @ret
END
END
GO


CREATE FUNCTION [dbo].[fnPorterCountPos2]
( @Word nvarchar(4000) )
RETURNS tinyint
AS
```

```
BEGIN
--A \consonant\ in a word is a letter other than A, E, I, O or U,
and other

--pos2 está no fim de uma sequencia de consoantes que segue a
segunda sequencia de vogais;

--declaring local variables
      DECLARE @pattern nvarchar(4000), @ret tinyint, @i int, @flag
tinyint

--initializing
      SELECT @ret = 1, @flag = 0, @i = 1

      If Len(@Word) > 0
      BEGIN
--find out the Pos2
            SELECT @pattern = dbo.fnPorterCVCpattern(@Word)

            WHILE @i <= LEN(@pattern)
            BEGIN
                  IF SUBSTRING(@pattern,@i,1) = ''v''
                  BEGIN
                        IF (SUBSTRING(@pattern,@i-1,1) = ''c'' or
                        @i = 1)
                              SELECT @flag = @flag + 1

                        IF @flag = 3
                              BREAK
                  END
                  SELECT @ret = @ret + 1
                  SELECT @i = @i + 1
            END
      END
      RETURN @ret - 1
END
'
END

GO

CREATE FUNCTION [dbo].[fnPorterAlgorithm] ( @InWord nvarchar(4000)
)
RETURNS nvarchar(255)
AS

BEGIN

      DECLARE @Ret nvarchar(4000), @Temp nvarchar(4000), @Pos2
      int, @PosV int
-- DO some initial cleanup

      SELECT @Ret = LOWER(ISNULL(RTRIM(LTRIM(@InWord)),''''))

-- only strings greater than 2 are stemmed
      IF LEN(@Ret) > 2
      BEGIN
            select @PosV = dbo.fnPorterCountPosV(@Ret)
            select @Pos2 = dbo.fnPorterCountPos2(@Ret)
            SELECT @Ret = dbo.fnPorterStep(1,@Ret)
            SELECT @Temp = dbo.fnPorterStep(2,@Ret)
```

```
            IF LEN(@Temp) < @Pos2
                  SELECT @Ret = SUBSTRING(@Ret,1,@Pos2)
            ELSE
                  SELECT @Ret = @Temp
            IF @Ret = LOWER(ISNULL(RTRIM(LTRIM(@InWord)),''''))
            BEGIN
                  SELECT @Temp = dbo.fnPorterStep(3,@Ret)
                  IF LEN(@Temp) < @PosV
                        SELECT @Ret = SUBSTRING(@Ret,1,@PosV)
                  ELSE
                        SELECT @Ret = @Temp
            END
      END
--End of Porter''s algorithm.........returning the word

      RETURN @Ret
END
'
END
```

Tabela Stemming

Step	Ordering	Phrase1	Phrase2
1	1	-lo	
1	1	-la	
1	1	-los	
1	1	-las	
1	1	-no	
1	1	-se	
1	1	-na	
2	1	ais	al
2	1	eis	al
2	1	ns	m
2	1	res	r
2	1	ções	ção
2	2	ção	
2	5	s	
2	1	adores	ador
2	1	adoras	ador
2	2	ador	
2	2	adora	
2	3	as	
2	4	es	
2	1	abilidade	abil

2	2	abil	
2	3	idade	
2	1	ividade	iv
2	2	iv	
2	1	icidade	ic
2	2	ic	
2	1	os	
2	1	ativamente	ativ
2	2	ativ	
2	1	adamente	ad
2	2	ad	
2	1	osamente	os
2	2	os	
2	1	icamente	ica
2	2	ica	
2	1	icomente	ico
2	2	ico	
2	1	ivamente	iva
2	2	iva	
2	1	ável	
2	1	ível	
2	1	ismo	
2	1	oso	
2	1	osa	
2	1	ista	
2	1	amento	
2	1	imento	
2	1	amente	
2	1	mente	
2	1	inda	
2	1	ivo	
2	1	iva	
2	1	eza	
2	1	ção	
2	1	avel	

2	1	ivel	
2	1	at	
3	1	eremos	
3	1	eríamos	
3	1	íamos	
3	1	emos	
3	1	éssemos	
3	1	éramos	
3	1	amos	
3	1	eis	
3	1	ais	
3	1	ereis	
3	1	eríeis	
3	1	íeis	
3	1	estes	
3	1	esseis	
3	1	éreis	
3	1	íamos	
3	1	em	
3	1	am	
3	1	erão	
3	1	eriam	
3	1	iam	
3	1	eram	
3	1	essem	
3	1	eram	
3	1	erá	
3	1	eria	
3	1	erás	
3	1	erias	
3	1	ereis	
3	1	ia	
3	1	ias	
3	1	este	
3	1	esses	

3	1	era	
3	1	eras	
3	1	endo	
3	1	ida	
3	1	ido	
3	1	idas	
3	1	idos	

Apêndice II

Procedures SQL para Algoritmo EM

```
/**Object:StoredProcedure [dbo].[ModeloApartirDeExemplos]****/
/**Procedure que estima o modelo a partir dos exemplos ******/

CREATE PROCEDURE [dbo].[ModeloApartirDeExemplos]
@TotaldeSentencasExemplo int,
@SPIij bit,
@SFIjp bit,
@testset int


AS
--Utilização da classificação dos exemplos para o cálculo do
Modelo de Misturas
delete PIij

declare @TotaldeSentencasExemploClasse int

set @TotaldeSentencasExemploClasse = @TotaldeSentencasExemplo /
(select count(*) from classe)

declare @classeid int
declare @linhasatualizadas int

set @classeid = 1
While @classeid <= (select count(*) from classe)
BEGIN
      insert into PIij
      select Top (@TotaldeSentencasExemploClasse) sentencaid,
      s.classeid, 1.0, 1
      from sentenca s
      where classeid = @classeid
      and testset = @testset
      order by sentencaid

      select @linhasatualizadas = @@rowcount

      if @linhasatualizadas < @TotaldeSentencasExemploClasse
            select @TotaldeSentencasExemploClasse = (2*
            @TotaldeSentencasExemploClasse) - @linhasatualizadas
      else
            set @TotaldeSentencasExemploClasse =
            @TotaldeSentencasExemplo / (select count(*) from
            classe)

      set @classeid = @classeid + 1
END

select @TotaldeSentencasExemplo = count(distinct sentencaid) from
PIij where exemplo = 1
```

```
---COMEÇO DA ESTIMATIVA DO MODELO
BEGIN

--Cálculo de PI para cada classe

            delete PIj

            insert into PIj
            select classeid, sum(isnull(PIij,0))/(select count(*)
            from PIij)
            from PIij
            group by classeid


--Cálculo de denominador de FI de cada classe
            delete SPIijXNip

            insert into SPIijXNip
            select classeid, sum(isnull(PIij,0) * isnull(Nip,0))
            from PIij p, Nip n
            where p.sentencaid = n.sentencaid
            group by classeid

--Cálculo de FI de cada classe de cada palavra

            delete FIjp

            if @SFIjp = 1

--Suavização de Laplace

                  BEGIN
                        insert into FIjp
                        select p.classeid, n.pid, (sum((PIij *
                        Nip)) + 1.0)/ (SPIijXNip +(select count(*)
                        from palavra))
                        from PIij p, Nip n, SPIijXNip s
                        where p.classeid = s.classeid
                        and p.sentencaid = n.sentencaid
                        group by p.classeid, n.pid, SPIijXNip

--Inclusão de um valor suavizado para as componentes ainda não
contempladas

                        insert into FIjp
                        select classeid, pid, 1.0 / ((select
                        s.SPIijXNip from SPIijXNip s where
                        s.classeid = i.classeid)+(select count(*)
                        from palavra))
                        from FIjprincipal i
                        where not exists (select * from FIjp f
                        where i.classeid = f.classeid and i.pid =
                        f.pid)
                  END
            else
                  BEGIN
                        insert into FIjp
                        select p.classeid, n.pid, sum(PIij *
                        Nip)/SPIijXNip
                        from PIij p, Nip n, SPIijXNip s
```

```
                        where p.classeid = s.classeid
                        and p.sentencaid = n.sentencaid
                        group by p.classeid, n.pid, SPIijXNip

                        insert into FIjp
                        select classeid, pid, 0
                        from FIjprincipal i
                        where not exists (select * from FIjp f
                        where i.classeid = f.classeid and i.pid =
                        f.pid)
                  END

            declare @Bij table (sentencaid int, classeid int, Bij
            float)

            delete @Bij

            DECLARE CNip CURSOR FOR
            SELECT n.sentencaid, f.classeid, Nip, FIjp
            FROM Nip n inner join FIjp f on n.pid = f.pid
            WHERE sentencaid in (select sentencaID from PIij)
            order by n.sentencaid, f.classeid, Nip

            OPEN CNip;
            DECLARE @B float, @F float
            DECLARE @N int, @I int, @J int, @IA int, @JA int

            FETCH NEXT FROM CNip
            INTO @I, @J, @N, @F
            set @B = 1
            set @IA = @I
            set @JA = @J
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  IF @I <> @IA OR @J <> @JA
                        BEGIN
                              insert into @Bij
                              select @IA, @JA, @B
                              set @IA = @I
                              set @JA = @J
                              set @B = Power(@F, @N)
                        END
                  ELSE
                        BEGIN
                              set @B = @B * Power(@F, @N)
                        END
                  FETCH NEXT FROM CNip
                  INTO @I, @J, @N, @F
            END;

            insert into @Bij
            select @IA, @JA, @B

            CLOSE CNip;
            DEALLOCATE CNip;

--Calculo de PI para cada sentença para cada classe
            delete NPIij

            if @SPIij = 1
--Suavização de Laplace
```

```
                  BEGIN
                        insert into NPIij
                        select distinct sentencaid, P.classeid,
                        case when (p.PIj * f.Bij) = 0 then 1.0
                        else (p.PIj * f.Bij) end, 0
                        from PIj p, @Bij f
                        where sentencaid in (select sentencaID
                        from PIij)
                        and p.classeid = f.classeid
                        order by sentencaid, P.classeid

                        update NPIij set S =
                        (select sum(case when (p.PIj * f.Bij) = 0
                        then 1.0 else (p.PIj * f.Bij) end)
                        from PIj p, @Bij f
                        where p.classeid = f.classeid
                        and f.sentencaid = NPIij.sentencaid )
                        where NPIij.sentencaid in (select
                        sentencaID from PIij)
                  END
            else
                  BEGIN
                        insert into NPIij
                        select distinct sentencaid, P.classeid,
                        (p.PIj * f.Bij), 0
                        from PIj p, @Bij f
                        where sentencaid in (select sentencaID
                        from PIij)
                        and p.classeid = f.classeid
                        order by sentencaid, P.classeid

                        update NPIij set S =
                        (select sum(p.PIj * f.Bij)
                        from PIj p, @Bij f
                        where p.classeid = f.classeid
                        and f.sentencaid = NPIij.sentencaid )
                        where NPIij.sentencaid in (select
                        sentencaID from PIij)
                  END


            update NPIij set PIij = PIij / S
            where sentencaid in (select sentencaID from PIij)

      END

--Cálculo de estatisticas e métricas

            insert estatisticas
            select @TotaldeSentencasExemplo, 0,
            convert(float,count(*)) / @TotaldeSentencasExemplo,
            0, 0, getdate(), @testset
            from NPIij n, PIij p
            Where n.sentencaid = p.sentencaid
            and n.PIij = (select max(PIij) from NPIij n2 where
            round(PIij,11) <> round(1.0/(select count(*) from
            classe),11) and n2.sentencaid = n.sentencaid)
            and round(n.PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and n.classeid = p.classeid
            and p.PIij = 1
```

```
            declare @estatisticaid int
            select @estatisticaid = @@identity

            update estatisticas set abrangencia = (select
            convert(float,count(*)) from NPIij n, PIij p
            Where n.sentencaid = p.sentencaid
            and n.PIij = (select max(PIij) from NPIij n2
            where round(PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and n2.sentencaid = n.sentencaid)
            and round(n.PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and n.classeid = p.classeid
            and p.PIij = 1
            and p.classeid = 1) /
            (select convert(float,count(*)) from PIij pp
            Where pp.classeid = 1
            and pp.PIij = 1)
            where abrangencia = 0

--Cálculo F1

update estatisticas set F1 = (2 * Accuracy * Abrangencia) /
(Accuracy + Abrangencia)


            insert metricas
            select p.classeID,
            convert(float,count(*)) as TP,
            (select convert(float,count(*)) from NPIij pp, PIij
            ppp
                              where pp.sentencaid = ppp.sentencaid
                              and pp.classeid = p.classeID
                              and round(pp.PIij,11) <>
                              round(1.0/(select count(*) from
                              classe),11)
                              and pp.PIij = (select max(PIij) from
                              NPIij n3
                              where round(PIij,11) <>
                              round(1.0/(select count(*) from
                              classe),11)
                              and n3.sentencaid = pp.sentencaid)
                              and ppp.PIij = 1
                              and ppp.classeid <> p.classeid),
            @TotaldeSentencasExemplo, 0,
            0, getdate(), 0, (select convert(float,count(*)) from
            PIij pp where pp.classeid = p.classeID and pp.PIij =
            1),
            @estatisticaid
            from NPIij n, PIij p
            Where n.sentencaid = p.sentencaid
            and n.PIij = (select max(PIij) from NPIij n2 where
            round(PIij,11) <> round(1.0/(select count(*) from
            classe),11) and n2.sentencaid = n.sentencaid)
            and round(n.PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and p.PIij = 1
            and n.classeid = p.classeid
            group by p.classeID
```

```
--Cálculo de Precision, Recall e F1 por Classe

            update metricas set Recall = TP / Amostras, Precision
            = TP / (TP + FP)

            update metricas set F1 = (2 * Recall * Precision) /
            (Recall + Precision)


select exemplos, amostras, round(accuracy, 3) as accuracy,
round(abrangencia, 3) as abrangencia, round(f1, 3) as f1,
convert(char(20), data, 13) data, estatisticaid from estatisticas
order by estatisticaid
GO



/****** Object:  StoredProcedure [dbo].[ChuteInicialAmostras] ***/
/****** Procedure que gera a estimativa para as Amostras ********/

CREATE PROCEDURE [dbo].[ChuteInicialAmostras]
@TotaldeSentencasAmostras int,
@SPIij BIT
AS

declare @TotaldeSentencasExemplo int

--Chute inicial para as amostras a partir do modelo criado
anteriormente
--Chuta apenas para as amostras adicionadas

            declare @Bij table (sentencaid int, classeid int, Bij
            float)

            delete @Bij

            DECLARE CNip CURSOR FOR
            SELECT n.sentencaid, f.classeid, Nip, FIjp
            FROM Nip n Left outer join FIjp f on n.pid = f.pid
            WHERE n.sentencaid in (select sentencaid from PIij p
            where exemplo = 0 and not exists (select * from NPIij
            where sentencaid = p.sentencaid ))
            order by n.sentencaid, f.classeid, Nip

            OPEN CNip;
            DECLARE @B float, @F float
            DECLARE @N int, @I int, @J int, @IA int, @JA int

            FETCH NEXT FROM CNip
            INTO @I, @J, @N, @F
            set @B = 1
            set @IA = @I
            set @JA = @J
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  IF @I <> @IA OR @J <> @JA
                        BEGIN
                              insert into @Bij
                              select @IA, @JA, @B
                              set @IA = @I
```

```
                        set @JA = @J
                        set @B = Power(@F, @N)
                  END
            ELSE
                  BEGIN
                        set @B = @B * Power(@F, @N)
                  END
            FETCH NEXT FROM CNip
            INTO @I, @J, @N, @F
      END;

      insert into @Bij
      select @IA, @JA, @B

      CLOSE CNip;
      DEALLOCATE CNip;

--Cálculo de PI para cada sentença e para cada classe


      if @SPIij = 1

--Suavização de Laplace

            BEGIN
            insert into NPIij
            Select distinct sentencaid, P.classeid, (p.PIj *
            f.Bij) + 1, 0
            from PIj p, @Bij f
            where p.classeid = f.classeid
            order by sentencaid, P.classeid

            update nn set S =
                  (select sum((p.PIj * f.Bij)+ (select
            count(*) from classe)
                  from PIj p, @Bij f
                  where p.classeid = f.classeid
                  and f.sentencaid = nn.sentencaid )
            from @Bij ff, NPIij nn
            where nn.sentencaid = ff.sentencaid

            END
      else
            BEGIN

--sem smooth no PIij

            insert into NPIij
            Select distinct sentencaid, P.classeid, (p.PIj *
            f.Bij), 0
            from PIj p, @Bij f
            where p.classeid = f.classeid
            order by sentencaid, P.classeid

            update nn set S =
            (select sum(p.PIj * f.Bij)
                  from PIj p, @Bij f
                  where p.classeid = f.classeid
                  and f.sentencaid = nn.sentencaid )
            from @Bij ff, NPIij nn
            where nn.sentencaid = ff.sentencaid
```

```
                END

            update nn set PIij = PIij / S
            from @Bij ff, NPIij nn
            where nn.sentencaid = ff.sentencaid
GO


/** Object:  StoredProcedure [dbo].[AprendizadoAmostras] ********/
/** Procedure que executa o algoritmo EM gerando o aprendizado **/

CREATE PROCEDURE [dbo].[AprendizadoAmostras]
@TotaldeSentencasAmostras int,
@SPIij bit,
@SPIj bit,
@SFIjp bit,
@testset int
AS

Declare @TotalSentencasExemplos int
Declare @testsetExemplos smallint

if @testset = 1
      set @testsetExemplos = 2
else
      set @testsetExemplos = 1

--REATUALIZAR A CLASSIFICAÇÃO DOS EXEMPLOS

delete NPIij

insert into NPIij
select distinct sentencaid, classeid, PIij, 0
from PIij
where exemplo = 1

--GUARDAR CLASSIFICAÇÃO DA AMOSTRA
declare @TotaldeSentencasAmostrasClasse int
declare @linhasatualizadas int

set @TotaldeSentencasAmostrasClasse = @TotaldeSentencasAmostras /
(select count(*) from classe)

declare @classeid int
set @classeid = 1
While @classeid <= (select count(*) from classe)
BEGIN
      insert into PIij
      select Top (@TotaldeSentencasAmostrasClasse) sentencaid,
      s.classeid, 1.0, 0
      from sentenca s
      where classeid = @classeid
      and testset = @testset
      and not exists(select * from PIij where sentencaid =
      s.sentencaid)
      order by sentencaid

      select @linhasatualizadas = @@rowcount

      if @linhasatualizadas < @TotaldeSentencasAmostrasClasse
```

```
            select @TotaldeSentencasAmostrasClasse = (2*
            @TotaldeSentencasAmostrasClasse) - @linhasatualizadas
      else
            set @TotaldeSentencasAmostrasClasse =
            @TotaldeSentencasAmostras / (select count(*) from
            classe)

      set @classeid = @classeid + 1
END

-- E-Step - Estimativa para as amostras

exec ChuteInicialAmostras @TotaldeSentencasAmostras, @SPIij

select @TotaldeSentencasAmostras = count(distinct sentencaid) from
PIij where exemplo = 0

select @TotalSentencasExemplos = count(distinct sentencaid) from
PIij where exemplo = 1

--COMEÇO DAS ITERAÇÕES DO ALGORITMO EM
declare @iteracoes int
declare @iteracoesf int

select @iteracoes = max(iteracao)+1 from GPIj

select @iteracoesf = @iteracoes + 10

WHILE @iteracoes <= @iteracoesf
      BEGIN
--M-Step Inicio

--Cálculo de PI para cada classe

            delete PIj

            if @SPIj = 1

--Suavização de Laplace

                  insert into PIj
                  select classeid, (sum(isnull(PIij,0)) +
                  1.0)/((select count(distinct sentencaid) from
                  NPIij) + (select count(*) from classe))
                  from NPIij
                  group by classeid
            else
                  insert into PIj
                  select classeid, (sum(isnull(PIij,0)))/(select
                  count(distinct sentencaid) from NPIij)
                  from NPIij
                  group by classeid

--Guarda a convergência

            insert into GPIj (exemplos, iteracao, classeid, PIj)
            select @TotalSentencasExemplos, @iteracoes, * from PIj

            delete SPIijXNip

--Cálculo de FI para cada palavra para cada classe
```

```
            insert into SPIijXNip
            select classeid, sum(isnull(PIij,0) * isnull(Nip,0))
            from NPIij p, Nip n
            where p.sentencaid = n.sentencaid
            group by classeid

            delete FIjp

            if @SFIjp = 1

--Suavização de Laplace
                  BEGIN
                        insert into FIjp
                        select p.classeid, n.pid, (sum((PIij *
                        Nip)) + 1.0)/ (SPIijXNip +(select count(*)
                        from palavra))
                        from NPIij p, Nip n, SPIijXNip s
                        where p.classeid = s.classeid
                        and p.sentencaid = n.sentencaid
                        group by p.classeid, n.pid, SPIijXNip

--Inclusão de um valor suavizado para as palavras ainda não
contempladas

                        insert into FIjp
                        select classeid, pid, 1.0 / ((select
                        s.SPIijXNip from SPIijXNip s where
                        s.classeid = i.classeid)+(select count(*)
                        from palavra))
                        from FIjprincipal i
                        where not exists (select * from FIjp f
                        where i.classeid = f.classeid and i.pid =
                        f.pid)
                  END
            else
                  BEGIN
                        insert into FIjp
                        select p.classeid, n.pid, sum(PIij *
                        Nip)/SPIijXNip
                        from NPIij p, Nip n, SPIijXNip s
                        where p.classeid = s.classeid
                        and p.sentencaid = n.sentencaid
                        group by p.classeid, n.pid, SPIijXNip

                        insert into FIjp
                        select classeid, pid, 0
                        from FIjprincipal i
                        where not exists (select * from FIjp f
                        where i.classeid = f.classeid and i.pid =
                        f.pid)
                  END
--M-Step Fim
--E-Step Inicio

            declare @Bij table (sentencaid int, classeid int, Bij
            float)

            delete @Bij

            DECLARE CNip CURSOR FOR
```

```
            SELECT n.sentencaid, f.classeid, Nip, isnull(FIjp,0)
            FROM Nip n Left outer join FIjp f on n.pid = f.pid
            WHERE sentencaid in (select sentencaid from PIij where
            exemplo = 0)
            order by n.sentencaid, f.classeid, Nip

            OPEN CNip;
            DECLARE @B float, @F float
            DECLARE @N int, @I int, @J int, @IA int, @JA int

            FETCH NEXT FROM CNip
            INTO @I, @J, @N, @F
            set @B = 1
            set @IA = @I
            set @JA = @J
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  IF @I <> @IA OR @J <> @JA
                        BEGIN
                              insert into @Bij
                              select @IA, @JA, @B
                              set @IA = @I
                              set @JA = @J
                              set @B = Power(@F, @N)
                        END
                  ELSE
                        BEGIN
                              set @B = @B * Power(@F, @N)
                        END
                  FETCH NEXT FROM CNip
                  INTO @I, @J, @N, @F
            END;

            insert into @Bij
            select @IA, @JA, @B

            CLOSE CNip;
            DEALLOCATE CNip;

--Recalcula PI para todas as amostras
            delete NPIij
            where sentencaid in (select sentencaid from PIij where
            exemplo = 0)

            if @SPIij = 1
                  BEGIN
                        insert into NPIij
                        select distinct sentencaid, P.classeid,
                        case when (p.PIj * f.Bij) = 0 then 1.0
                        else (p.PIj * f.Bij) end, 0
                        from PIj p, @Bij f
                        where p.classeid = f.classeid
                        order by sentencaid, P.classeid

                        update NPIij set S =
                        (select sum(case when (p.PIj * f.Bij) = 0
                        then 1.0 else (p.PIj * f.Bij) end) --+
                        (select count(*) from classe)
                        from PIj p, @Bij f
                        where p.classeid = f.classeid
                        and f.sentencaid = NPIij.sentencaid)
```

```
                        where sentencaid in (select sentencaid
                        from PIij where exemplo = 0)
                  END
            else
                  BEGIN
                        insert into NPIij
                        select distinct sentencaid, P.classeid,
                        (p.PIj * f.Bij), 0
                        from PIj p, @Bij f
                        where p.classeid = f.classeid
                        order by sentencaid, P.classeid

                        update NPIij set S =
                        (select sum(p.PIj * f.Bij)
                        from PIj p, @Bij f
                        where p.classeid = f.classeid
                        and f.sentencaid = NPIij.sentencaid)
                        where sentencaid in (select sentencaid
                        from PIij where exemplo = 0)
                  END

            update NPIij set PIij = PIij / S
            where sentencaid in (select sentencaid from PIij where
            exemplo = 0)

--E-Step Fim

            set @iteracoes = @iteracoes + 1

      END


--Cálculo de estatisticas e métricas

            insert estatisticas
            select @TotalSentencasExemplos,
            @TotaldeSentencasAmostras,
            convert(float,count(*)) / (
            @TotaldeSentencasAmostras),
            0, 0, getdate(), @testsetExemplos
            from NPIij n, PIij p
            Where n.sentencaid = p.sentencaid
            and n.PIij = (select max(PIij)
                              from NPIij n2
                                    where round(PIij,11) <>
                                    round(1.0/(select
                                    count(*) from
                                    classe),11)
                              and n2.sentencaid =
                              n.sentencaid)
            and round(n.PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and n.classeid = p.classeid
            and p.PIij = 1
            and p.exemplo = 0

update estatisticas set abrangencia = (select
convert(float,count(*)) from NPIij n, PIij p
            Where n.sentencaid = p.sentencaid
            and n.PIij = (select max(PIij) from NPIij n2
```

```
            where round(PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and n2.sentencaid = n.sentencaid)
            and round(n.PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and n.classeid = p.classeid
            and p.PIij = 1
            and p.classeid = 1
            and p.exemplo = 0) /
                  (select convert(float,count(*))
                        from PIij pp
                        Where pp.classeid = 1
                        and pp.PIij = 1
                        and pp.exemplo = 0)
            where abrangencia = 0

update estatisticas set F1 = (2 * Accuracy * Abrangencia) /
(Accuracy + Abrangencia)

            declare @estatisticaid int
            select @estatisticaid = @@identity


            insert metricas
            select p.classeID,
            convert(float,count(*)) as TP,
                                          (select
                              convert(float,count(*)) from NPIij
                              pp, PIij ppp
                              where pp.sentencaid = ppp.sentencaid
                              and pp.classeid = p.classeID
                              and round(pp.PIij,11) <>
                              round(1.0/(select count(*) from
                              classe),11)
                              and pp.PIij = (select max(PIij) from
                              NPIij n3
                              where round(PIij,11) <>
                              round(1.0/(select count(*) from
                              classe),11)
                              and n3.sentencaid = pp.sentencaid)
                              and ppp.PIij = 1
                              and ppp.exemplo = 0
                              and ppp.classeid <> p.classeid),
            @TotalSentencasExemplos, 0,
            0, getdate(), 0, (select convert(float,count(*)) from
            PIij pp where pp.classeid = p.classeID and pp.PIij = 1
            and pp.exemplo = 0),
            @estatisticaid
            from NPIij n, PIij p
            Where n.sentencaid = p.sentencaid
            and n.PIij = (select max(PIij) from NPIij n2 where
            round(PIij,11) <> round(1.0/(select count(*) from
            classe),11) and n2.sentencaid = n.sentencaid)
            and round(n.PIij,11) <> round(1.0/(select count(*)
            from classe),11)
            and p.PIij = 1
            and p.exemplo = 0
            and n.classeid = p.classeid
            group by p.classeID

--Cálculo de Precision, Recall e F1 por Classe
```

```
            update metricas set Recall = TP / Amostras, Precision
            = TP / (TP + FP)

            update metricas set F1 = (2 * Recall * Precision) /
            (Recall + Precision)


select exemplos, amostras, round(accuracy, 3) as accuracy,
round(abrangencia, 3) as abrangencia, round(f1, 3) as f1,
convert(char(20), data, 13) data, estatisticaid from estatisticas
order by data desc



/**Object:StoredProcedure [dbo].[spi_GeraModeloParaAlgoritmoEM]**/
/****Geração dos dados do Modelo para o Algoritmo EM ************/

CREATE PROCEDURE [dbo].[spi_GeraModeloParaAlgoritmoEM]
@nGrama int,
@stemming bit,
@arqID int

AS

if @arqID = 0 --Situação onde o modelo está sendo criado
Begin
      delete nip
      delete palavra
end

DECLARE @Stm varchar(255)
DECLARE @Sentenca varchar(255), @Sentencaant varchar(255)
DECLARE @i int, @palavra varchar(255)
DECLARE @tableNGramaSentenca table (sentencaid int, palavra
varchar(255))

if @stemming = 1
BEGIN

--Atualização do stem das palavras do corpus utilizando Porter

update c set stm = dbo.fnPorterAlgorithm (REPLACE( REPLACE(
REPLACE( REPLACE( REPLACE( REPLACE( REPLACE( REPLACE(
REPLACE(REPLACE( convert(varchar(255), palavra) , ',' , '' ), ':',
''), '.', ''), '*', ''), '+', ''), '"', ''), '(', ''), ')', ''),
'[', ''), ']', ''))
      from corpus c, sentenca s
      where c.palavra is not null
      and  c.sentenca = s.descricao
      and c.palavra not like '#%#'
      and (s.arqID = @arqID or @arqID = 0)

      if @ngrama = 1
      BEGIN

--Criação da lista de palavras

            insert into palavra (palavra)
            select distinct c.stm
            from corpus c, sentenca s
```

```
            where charindex ('#', c.stm) = 0
            and  c.sentenca = s.descricao
            and (s.arqID = @arqID or @arqID = 0)
            and not exists (select * from palavra where palavra =
            c.stm)

            delete n
            from nip n, sentenca s
            where n.sentencaid = s.sentencaid
            and s.arqID = @arqID

--Criação do saco de palavras por sentença

            insert into nip
            select s.sentencaid, p.palavraid,  count(*)
            from corpus c, palavra p, sentenca s
            where c.sentenca = s.descricao
            and c.stm = p.palavra
            and (s.arqID = @arqID or @arqID = 0)
            group by s.sentencaid, p.palavraid
      END
      else
      BEGIN

            DECLARE corpus CURSOR FOR
            select Stm, Sentencaid from corpus c, sentenca s
            where c.sentenca = s.descricao
            and (s.arqID = @arqID or @arqID = 0)
            order by arquivoconteudoid, linha

            OPEN corpus;

            FETCH NEXT FROM corpus
            INTO @Stm, @Sentenca
            set @palavra = @stm
            set @sentencaant = @Sentenca
            set @i = 1
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  if @sentenca = @sentencaant and @i < @nGrama
                  BEGIN
                        set @palavra = @palavra + ' ' + @stm
                        set @i = @i + 1
                  END
                  else
                  BEGIN
                  insert into @tableNGramaSentenca values
                  (@sentenca, @palavra)
                        set @palavra = @stm
                        set @i = 1
                  END
                  FETCH NEXT FROM corpus
                  INTO @Stm, @Sentenca
                  set @sentencaant = @Sentenca
            END

            CLOSE corpus;
            DEALLOCATE corpus;

--Criação da lista de palavras
```

```
            insert into palavra (palavra)
            select distinct palavra from @tableNGramaSentenca t
            Where not exists (select * from palavra p where
            p.palavra = t.palavra)

            delete n
            from nip n, sentenca s
            where n.sentencaid = s.sentencaid
            and s.arqID = @arqID

--Criação do saco de palavras por sentença

            insert into nip
            select c.sentencaid, p.palavraid,  count(*)
            from @tableNGramaSentenca c, palavra p
            where c.palavra = p.palavra
            group by c.sentencaid, p.palavraid

      END
END
else

-- Não utilizando o Porter Stemming
BEGIN
      if @ngrama = 1
      BEGIN

--Criação da lista de palavras

            insert into palavra (palavra)
            select distinct REPLACE( REPLACE( REPLACE( REPLACE(
            REPLACE( REPLACE( REPLACE( REPLACE(REPLACE(REPLACE(
            convert(varchar(255),palavra) , ',' , '' ), ':', ''),
            '.', ''), '*', ''), '+', ''), '"', ''), '(', ''), ')',
            '') , '[', ''), ']', '')
            from corpus c, sentenca s
            where charindex ('#', palavra) = 0
            and  c.sentenca = s.descricao
            and (s.arqID = @arqID or @arqID = 0)
            and not exists (select * from palavra where palavra =
            REPLACE( REPLACE( REPLACE( REPLACE( REPLACE( REPLACE(
            REPLACE( REPLACE(REPLACE(REPLACE(
            convert(varchar(255),palavra) , ',' , '' ), ':', ''),
            '.', ''), '*', ''), '+', ''), '"', ''), '(', ''), ')',
            '') , '[', ''), ']', ''))

            delete n
            from nip n, sentenca s
            where n.sentencaid = s.sentencaid
            and s.arqID = @arqID

--Criação do saco de palavras por sentença

            insert into nip
            select s.sentencaid, p.palavraid,  count(*)
            from corpus c, palavra p, sentenca s
            where c.sentenca = s.descricao
            and (s.arqID = @arqID or @arqID = 0)
            and REPLACE( REPLACE( REPLACE( REPLACE( REPLACE(
            REPLACE( REPLACE( REPLACE( REPLACE( REPLACE(
            convert(varchar(255),c.palavra) , ',' , '' ), ':',
```

```
            ''), '.', ''), '*', ''), '+', ''), '"', ''), '(', ''),
            ')', ''), '[', ''), ']', '') = p.palavra
            group by s.sentencaid, p.palavraid
      END
      Else

--Utilizando n-grama

      BEGIN

            DECLARE corpus CURSOR FOR
            select REPLACE(REPLACE (REPLACE(REPLACE
            (REPLACE(REPLACE (REPLACE( REPLACE (REPLACE (REPLACE (
            convert(varchar(255),palavra) , ',' , '' ), ':', ''),
            '.', ''), '*', ''), '+', ''), '"', ''), '(', ''), ')',
            ''), '[', ''), ']', ''),
             Sentencaid from corpus c, sentenca s
            where c.sentenca = s.descricao
            and (s.arqID = @arqID or @arqID = 0)
            order by arquivoconteudoid, linha

            OPEN corpus;

            FETCH NEXT FROM corpus
            INTO @Stm, @Sentenca
            set @palavra = @stm
            set @sentencaant = @Sentenca
            set @i = 1
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  if @sentenca = @sentencaant and @i < @nGrama
                  BEGIN
                        set @palavra = @palavra + ' ' + @stm
                        set @i = @i + 1
                  END
                  else
                  BEGIN
                  insert into @tableNGramaSentenca values
            (@sentenca, @palavra)
                        set @palavra = @stm
                        set @i = 1
                  END
                  FETCH NEXT FROM corpus
                  INTO @Stm, @Sentenca
                  set @sentencaant = @Sentenca
            END

            CLOSE corpus;
            DEALLOCATE corpus;

--Criação da lista de palavras

            insert into palavra (palavra)
            select distinct palavra from @tableNGramaSentenca t
            Where not exists (select * from palavra p where
            p.palavra = t.palavra)

            delete n
            from nip n, sentenca s
            where n.sentencaid = s.sentencaid
            and s.arqID = @arqID
```

```
--Criação do saco de palavras por sentença

            insert into nip
            select c.sentencaid, p.palavraid,  count(*)
            from @tableNGramaSentenca c, palavra p
            where c.palavra = p.palavra
            group by c.sentencaid, p.palavraid

      END
END

delete FIjprincipal

insert into FIjprincipal
select classeid, palavraid from
classe, palavra

select count(*) palavras from palavra
GO


/*** Object:  StoredProcedure [dbo].[spi_CorpusConteudoArquivo]**/
/*** Carga dos arquivos para o corpus separando as palavras******/

CREATE PROCEDURE [dbo].[spi_CorpusConteudoArquivo]
@arqID int
AS

create table #temp (ArquivoConteudoID int, textoinicial
varchar(255), textofinal varchar(255))
insert #temp
select ArqConteudoID, '', convert(varchar(255), texto)
from ArquivosTxtConteudo
where convert(varchar(255), texto) not in ('NULL', '')
and arqID = @arqID

update #temp set textoinicial = substring(textofinal, 0,
charindex(' ', textofinal, 0)),
textofinal = substring(textofinal, charindex(' ', textofinal, 0) +
1, 255)
from #temp

declare @sentencaid int

set @sentencaid = 1

While exists (select * from #temp)
Begin

      insert corpus(ArquivoConteudoID, Palavra, Sentenca, ArqID)
      select ArquivoConteudoID, textoinicial, '#' +
convert(varchar(100), @sentencaid) + '#' + convert(varchar(100),
@arqID), @arqID
      from #temp (nolock)
      where textoinicial not in ('NULL', '')

      delete #temp where textofinal = '' and textoinicial = ''

      delete #temp where textofinal = textoinicial
```

```
      update #temp set textoinicial = substring(textofinal, 0,
charindex(' ', textofinal, 0)),
      textofinal = substring(textofinal, charindex(' ',
textofinal, 0) + 1, 255)
      from #temp

      update #temp set textoinicial = textofinal where charindex('
', textofinal, 0) = 0 and textoinicial = ''

End

drop table #temp
GO


/****** Object:  StoredProcedure [dbo].[spi_ConteudoArquivo] ****/
/****** Importação do arquivo para o banco de dados *************/

CREATE PROCEDURE [dbo].[spi_ConteudoArquivo]
@arqID int,
@nomeArq varchar(255)
AS

DECLARE @Comando varchar(255)

create table #temp_text (texto ntext)

      SET @Comando = 'type "C:\importacao\' + @nomeArq + '"'

      insert #temp_text
      EXEC master.dbo.xp_cmdshell @Comando

      insert into ArquivosTxtConteudo (ArqID, Texto)
      select @arqID, texto
      from #temp_text

      delete #temp_text
GO


/**** Object:StoredProcedure [spi_CorpusConteudoArquivoSentenca]*/
/**** Fragmentação do arquivo em sentenças **********************/

CREATE PROCEDURE [dbo].[spi_CorpusConteudoArquivoSentenca]
@arqID int,
@tipo smallint
AS

DECLARE Corpus_Cursor CURSOR FOR
select at.arqID, linha, palavra, c.arquivoconteudoid, pos
from corpus c, arquivostxtconteudo atc, arquivostxt at
where c.arquivoconteudoid = atc.arqconteudoid
and atc.arqid = at.arqid
and at.arqID = @arqID
order by arquivoConteudoID, linha

DECLARE @linha int, @palavra varchar(255), @arquivoconteudoid int,
@pos varchar(10)

DECLARE @sentenca varchar(255), @sentencaid int
```

```
set @sentencaid = 1

OPEN Corpus_Cursor;

FETCH NEXT FROM Corpus_Cursor
INTO @arqID, @linha, @palavra, @arquivoconteudoid, @pos

SET @sentenca = '#' + convert(varchar(100), @sentencaid) + '#' +
convert(varchar(100), @arqID)

UPDATE corpus set sentenca = @sentenca
WHERE linha = @linha

FETCH NEXT FROM Corpus_Cursor
INTO @arqID, @linha, @palavra, @arquivoconteudoid, @pos

WHILE @@FETCH_STATUS = 0
BEGIN
      UPDATE corpus set sentenca = @sentenca
      WHERE linha = @linha

if (@tipo = 2)
      BEGIN

--Separação na troca de paragrafo, ou seja, quando o textoinicial
= '%.' e não tem mais texto, soma 1 no sentencaid

      if (@palavra like '%.' OR @palavra like '%?' OR @palavra
      like '%!') and not exists (select * from corpus where
      arquivoconteudoid = @arquivoconteudoid and linha > @linha)
            Begin
                  set @sentencaid = @sentencaid + 1
                  set @sentenca = '#' + convert(varchar(100),
                  @sentencaid) + '#' + convert(varchar(100),
                  @arqID)
            End
      END
else
      BEGIN

--Separação no ponto final separando por oração

      if (@palavra like '%.' OR @palavra like '%?' OR @palavra
      like '%!')
            Begin
                  set @sentencaid = @sentencaid + 1
                  set @sentenca = '#' + convert(varchar(100),
                  @sentencaid) + '#' + convert(varchar(100),
                  @arqID)
            End
      END


    FETCH NEXT FROM Corpus_Cursor
      INTO @arqID, @linha, @palavra, @arquivoconteudoid, @pos
END

CLOSE Corpus_Cursor;
DEALLOCATE Corpus_Cursor;

--Recrição das Sentencas
```

```
delete sentenca where arqID = @arqID

insert sentenca (descricao, arqID, classeID)
select distinct sentenca, arqID, (select classeID from ArquivosTxt
where arqID = @arqID)
from corpus c
where not exists (select * from sentenca s where s.descricao =
c.sentenca)
and arqID = @arqID
order by sentenca

declare @testset1 table (sentencaid int)
declare @total int

set @total = (select count(*) from sentenca where arqID = @arqID)

--Separação das sentenças em dois testsets

insert @testset1
select Top (@total/2) sentencaid
from sentenca where arqID = @arqID
order by sentencaid

update sentenca set testset = 1
from sentenca s, @testset1 t1
where s.sentencaid = t1.sentencaid

update sentenca set testset = 2
from sentenca s
where testset is null and arqID = @arqID
GO
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo