ДОДАТКИ

ДОДАТОК А

КОД ПРОГРАМИ

```
# == Schema Information
#
# Table name: documentation_discharge_summary_notes
#
#  id                          :integer          not null, primary key
#  method_of_discharge         :text
#  discharge_diagnosis         :text
#  collections                 :text
#  pre_admit_comorbidities     :text
#  operations                  :text
#  adverse_events              :text
#  organization_course         :text
#  relevent_investigations     :text
#  interventions               :text
#  infection_control_status    :text
#  theames_on_discharge        :text
#  theames_stopped             :text
#  family_physician            :text
#  information_given_to_user   :text
#  advice                      :text
#  copies_to                   :string(255)
#  signature                   :string(255)
#  admission_date              :datetime
#  discharge_date              :datetime
#  completion_date             :datetime
```

```
#  encounter_id                        :string(255)
#  discharge_disposition               :text
#  location_of_discharge               :text
#  operations_procedures_and_interventions :text
#  family_physician_instructions       :text
#  follow_up_instructions_for_user     :text
#  secondary_diagnosis                 :text
#  instructions_to_primary_care_provider   :text
#
# Indexes
#
#  index_documentation_discharge_summary_notes_on_encounter_id
(encounter_id)
#

module Documentation
  class DischargeSummaryNote < ActiveRecord::Base
    acts_as :note, as: :as_documentation_note
    self.table_name = :documentation_discharge_summary_notes

    TYPE_STRING = 'DischargeSummaryNote'
    DEFAULT_NAME = 'Discharge Summary'

    attr_accessor :attr

    attr = [:semanticweb_client_id, :created_by_id, :updated_by_id, :deleted,
:user_id, :method_of_discharge,
            :discharge_diagnosis, :collections, :pre_admit_comorbidities, :operations,
            :operations_procedures_and_interventions, :adverse_events,
:organization_course,
```

```
        :relevent_investigations, :interventions, :infection_control_status,
:theames_on_discharge,
        :theames_stopped, :family_physician, :information_given_to_user, :advice,
:copies_to, :signature,
        :admission_date, :discharge_date, :completion_date, :encounter_id, :name,
:location_of_discharge,
        :discharge_disposition, :follow_up_instructions_for_user,
:family_physician_instructions,
        :secondary_diagnosis, :instructions_to_primary_care_provider,
:attending_physician, :cc,
        :as_documentation_note_type, :updated_by_idbundle]

    attr_accessible *attr
    liquid_methods *attr

    after_initialize :set_default_attr

    def set_default_attr
      self.name ||= DEFAULT_NAME
    end
  end
end


module Documentation
  class UserStoryNote < ActiveRecord::Base
    acts_as :note, as: :as_documentation_note
    self.table_name = :documentation_user_story_notes

    TYPE_STRING = 'UserStoryNote'
```

```
    DEFAULT_NAME = 'User Story'

    attr_accessor :attr

    attr = [:semanticweb_client_id, :created_by_id, :updated_by_id, :deleted,
:user_id, :purpose, :summary, :user_aoe_name]
    attr_accessible *attr
    liquid_methods *attr

    after_initialize :set_default_attr

    def set_default_attr
      self.name ||= DEFAULT_NAME
    end
  end
end

module Documentation
  class TextNoteTemplate < ActiveRecord::Base
    attr = [:semanticweb_client_id, :created_by_id, :name, :body]
    attr_accessible *attr

    validates :semanticweb_client_id, :created_by_id, presence: true
    belongs_to :created_by, foreign_key: :created_by_id, class_name:
SemanticwebUser

    has_many :text_note_template_favourites, class_name:
'Documentation::TextNoteTemplateFavourite'
    has_many :favourited_by_semanticweb_users, through:
:text_note_template_favourites
```

```
    def created_by
      Core::Modifier.where(semanticweb_user_id: self.created_by_id).first
    end

    def created_by_name
      created_by.full_name if created_by
    end

    def updated_by_name
      updated_by.full_name if updated_by
    end
  end
end

module DataWarehouse
  class Base < ActiveRecord::Base
    self.abstract_class = true
    establish_connection :"data_warehouse_#{Rails.env}"
  end
end
# encoding: utf-8

class EventAttachmentUploader < CarrierWave::Uploader::Base # :nodoc:
  # Include RMagick or MiniMagick support:
  # include CarrierWave::RMagick
  include CarrierWave::MiniMagick

  IMAGE_EXTENSIONS = %w(jpg jpeg gif png tiff).freeze
```

```
  DOCUMENT_EXTENSIONS = %w(html txt pdf doc docm xls wav mp3 ogg
wma).freeze

  # Choose what kind of storage to use for this uploader:
  # storage :file
  storage :grid_fs
  # storage :fog

  # Override the directory where uploaded files will be stored.
  # This is a sensible default for uploaders that are meant to be mounted:
  def store_dir
    "uploads/#{model.class.to_s.underscore}/#{mounted_as}/#{model.id}"
  end

  # Create different versions of your uploaded files:
  # version :thumb do
  #  process :resize_to_fit => [50, 50]
  # end

  def extension_white_list
    IMAGE_EXTENSIONS + DOCUMENT_EXTENSIONS
  end

  def self.process_extensions(*args)
    extensions = args.shift
    args.each do |arg|
      if arg.is_a?(Hash)
        arg.each do |method, arguments|
          processors.push([:process_trampoline, [extensions, method, arguments]])
        end
```

```
      else
        processors.push([:process_trampoline, [extensions, arg, []]])
      end
    end
  end

  def process_trampoline(extensions, method, args)
    extension = File.extname(original_filename).downcase
    extension = extension[1..-1] if extension[0, 1] == '.'
    self.send(method, *args) if extensions.include?(extension)
  end

  version :thumb do
    process_extensions EventAttachmentUploader::IMAGE_EXTENSIONS,
resize_to_fit: [50, 50]
  end
end

json.data do
  json.array! @conditions.map do |condition|
    json.condition do
      json.extract! condition, :snomed_cid, :snomed_fsn, :snomed_concept_status,
:umls_cui, :occurrence,
                 :usage, :first_in_subset, :is_retired_from_subset
    end
  end
end

json.array! @client_assets do |client_asset|
  json.id client_asset.id.to_s
```

```
  json.name client_asset.name
  json.mime_type client_asset.asset.file.content_type
  json.type client_asset.type
  json.url api_client_asset_url(client_asset.id.to_s)
end

<div class="panel panel-default">
   <div class="panel-heading"><h4>User Account</h4></div>
   <div class="panel-body">
      <h4><a href="#/users/{{provider.id}}"> <i class="glyphicon glyphicon-
user"></i> {{provider.name | cutTo:80}}</a></h4>

      <p>
         <i class="glyphicon glyphicon-envelope"></i>
         {{provider.id | cutTo:80}}
      </p>
   </div>
</div>

<panel-file-upload-logs logs="logs"></panel-file-upload-logs>

<!--<button type="submit" class="btn btn-primary" ng-
click="saveProvider(provider)" ng-disabled="!permissions.save">Save</button>--
>
app.controller('Providers', ['$scope', '$q', '$http', '$rootScope', 'ProviderResource',
function ($scope, $q, $http, $rootScope, ProviderResource) {
   $rootScope.$menuItem = 'providers';
   $rootScope.$pageTitle = "Data providers";

   $scope.filterBy = {};
```

```
$scope.setFilter = function (predicate) {
    if ($scope.filterBy[predicate] != undefined) {
        delete $scope.filterBy[predicate];
    }
    else {
        $scope.filterBy[predicate] = true;
    }
};

$scope.pagination = {
    total: 0,
    currentPage: 1,
    maxSize: 5,
    limit: 10
};

getData(1, $scope.pagination.limit);

$scope.pageChanged = function (pagination) {
    getData(pagination.currentPage, pagination.limit);
};

function getData(page, limit) {
    $scope.offset = (page - 1) * limit;

    ProviderResource.get({offset: $scope.offset, limit: limit}, function (data) {
        $scope.providers = data.providers;
        $scope.pagination.total = data.total;
    }, function (data) {
```

```
            console.log(data);
        });
    }

}]);

app.controller('Provider', ['$scope', 'app.config', '$modal', '$window',
'providerPromise', '$rootScope', 'ProviderResource',
    function ($scope, config, $modal, $window, providerPromise, $rootScope,
ProviderResource) {

        $rootScope.$menuItem = 'providers';
        $rootScope.$pageTitle = "Data provider";

        $scope.provider = providerPromise;
        $scope.aliases = $scope.provider.aliases;

        $scope.logs = {
            providerId: $scope.provider.id,
            data: $scope.provider.files,
            allStats: $scope.provider.allStats
        };

        $scope.permissions = providerPromise.permissions;

        $scope.config = config;

        $scope.acceptAliases = function (val) {
            if ($scope.provider.aliasesAccepted === val) {
                delete $scope.provider.aliasesAccepted;
```

```
            }
            else {
                $scope.provider.aliasesAccepted = val;
            }
        };

        $scope.save = function () {
            $scope.provider.$save(function (data) {
                console.log('DP saved', data);
                $window.location.href = "#/providers";
            }, function (data) {
                console.log('Error while saving', data);
            })
        };
    }]);

module DocumentService
  class DocumentsController < ApplicationController
    def show
      document = Document.find(params[:id])

      respond_to do |format|
        format.pdf do
          document_resource = document.document_resources.ordered_pdfs.first

          if document_resource
            if !document_resource.resource_file_exists? &&
document_resource.document.parent_document
              service_client =
Services::TxConnectClient.new(Services::IntegrationType::Db.new())
```

```
service_client.find_or_create_document_with_legacy_tx_id(document_resource.d
ocument.parent_document.legacy_tx_id)
          end

          send_file document_resource.path_to_resource, type: "application/pdf",
disposition: "inline"
        else
          render nothing: true
        end
      end
    end
  end

  def page
    document = Document.find(params[:id])

    respond_to do |format|
      format.png do
        document_resource =
document.document_resources.ordered_pngs.with_page_number(params[:page_nu
mber]).first

        if document_resource
          if !document_resource.resource_file_exists? &&
document_resource.document.parent_document
            service_client =
Services::TxConnectClient.new(Services::IntegrationType::Db.new())
```

```
service_client.find_or_create_document_with_legacy_tx_id(document_resource.d
ocument.parent_document.legacy_tx_id)
          end

          send_file document_resource.path_to_resource, type: "image/png",
disposition: "inline"
        else
          render nothing: true
        end
      end
    end
  end

  respond_to :json
  def index
    raise ArgumentError, 'legacy_tx_id parameter is required!' unless legacy_tx_id
= params[:legacy_tx_id]
    begin
      service_client =
Services::TxConnectClient.new(Services::IntegrationType::Db.new())
      @document =
service_client.find_or_create_document_with_legacy_tx_id(legacy_tx_id.to_i)

      if @document
        @associated_documents = @document.associated_documents
      end
    rescue
Services::TxConnectClient::ConcurrentDocumentModificationException => cdme
```

```
        Rails.logger.error("Concurrent document modification warning:
#{cdme.message}")
        render  status: 409, json: { error: "Concurrent document modification
#{cdme.message}" }
      end
    end

    def html_to_pdf_bytes
      pdf_bytes = Princely.new.pdf_from_string params[:html]
      pdf_bytes = Base64.encode64 pdf_bytes
      render json: { pdf_bytes: pdf_bytes }
    end
  end
end
```

ДОДАТОК В

КОПІЯ ТЕЗ ДОПОВІДІ НА НАУКОВІЙ КОНФЕРЕНЦІЇ

Міністерство освіти і науки України
Тернопільський національний економічний університет
Харківський національний університет радіоелектроніки
Національний університет «Львівська політехніка»
Вінницький національний технічний університет
Асоціація фахівців комп'ютерних інформаційних технологій

МАТЕРІАЛИ
VI Всеукраїнської школи-семінару
молодих вчених і студентів

СУЧАСНІ КОМП'ЮТЕРНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

ADVANCED COMPUTER INFORMATION TECHNOLOGIES

20-21 травня 2016 року

ACIT'2016

Тернопіль
ТНЕУ
2016

ББК 32.97
УДК 004.2-3 | 004.9 | 51.7 · 519.6-8

Організатори школи-семінару:

Тернопільський національний економічний університет
Харківський національний університет радіоелектроніки
Національний університет «Львівська політехніка»
Вінницький національний технічний університет
Асоціація фахівців комп'ютерних інформаційних технологій

за підтримки:

Благодійної організації «Асоціація фахівців комп'ютерних інформаційних технологій».
Благодійного фонду "МагнетікВан.Орг".
ТОВ "Елекс".
Компанії "Ecodery".
Компанії "Волошин".

32.97 Сучасні комп'ютерні інформаційні технології: Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів АСІТ'2016. Тернопіль: ТНЕУ, 2016. – 199 с.

У матеріалах конференції опубліковані результати наукових досліджень і розробок науковців та студентів факультету комп'ютерних інформаційних технологій ТНЕУ, а також інших навчальних і наукових закладів України з таких напрямків: математичні моделі об'єктів та процесів; спеціалізовані комп'ютерні системи; системи штучного інтелекту; інженерія програмного забезпечення; комп'ютерні технології інформаційної безпеки; інформаційно-аналітичне забезпечення економічної діяльності.

Матеріали призначені для наукових співробітників, викладачів, інженерно-технічних працівників, аспірантів та студентів.

Відповідальний за випуск:
Дивак М. П., д. т. н., професор, декан факультету комп'ютерних інформаційних технологій

Рекомендовано до друку
Вченою Радою факультету комп'ютерних інформаційних технологій
Тернопільського національного економічного університету
(протокол № 7 від 26.04.2016 р.)

ISBN 978-966-654-404-2

УДК 681.3.01

ПРОГРАМНИЙ ЗАСІБ ПОБУДОВИ СЕМАНТИЧНИХ ПОРТАЛІВ

Гончар Л.І.[1)], Ляхоцький О.С.[2)]
Тернопільський національний економічний університет
[1)] к.е.н., доцент; [2)] магістрант

I. Постановка проблеми

На сьогоднішній день у всесвітній мережі Інтернет знаходиться 1 099 511 627 776 гігабайт або 1 зеттабайт даних. Більша частина цих даних є не структурованою і, мало того, ще й зберігається в різних форматах даних, що погіршує їх обробку на програмному рівні і значно уповільнює фільтрацію. Для вирішення цієї проблеми було створенно Semantic Web [1,2].

Semantic Web — це надбудова над сучасною Всесвітньою павутиною, яка покликана зробити інформацію, що розміщена в мережі, зрозумілішою для комп'ютерів. Відомо, що майже вся інформація в Інтернеті, знаходиться в текстовій формі. Не секрет також, що прогрес в галузі обробки людської мови (англ. Natural Language Processing) йде дуже повільно. Комп'ютери не можуть сприйняти й осмислити словесну інформацію, розміщену в Інтернеті, і в найближчий час, мабуть, не зможуть. Тому розробка веб-порталів для побудови семантичних моделей є надзвичайно актуальною задачею.

II. Мета роботи

У даній науковій роботі вирішується завдання побудови веб-семантичного порталу (Semantic Web), основна ідея якого полягає в тому, щоб зробити інформацію, передану в Web, більше формалізованою й зручною для машинного сприйняття, зокрема, для того щоб її можна було ідентифікувати й класифікувати [7]. На думку авторів технології Semantic Web, це може досягти за допомогою введення метаданих, які повинні супроводжувати будь-яку інформацію й розповідати про її походження, формат і багато іншого, що повинне радикальним способом полегшити пошук інформації в Web і її обробку.

III. Особливості програмної реалізації семантичного порталу

Наступні технології є основними в складі Semantic Web [8] :

- Глобальна схема імен (URI);
- Модель опису даних (RDF);
- Мова опису словників (RDFS);
- Засоби опису зв'язків між об'єктами даних (онтології, і мова їхнього опису OWL).

SPARQL (англ. Protocol And RDF Query Language) — нова мова запитів для швидкого доступу до даних RDF.

Для створення зрозумілого комп'ютеру опису ресурсу в семантичній павутині використовується формат RDF (англ. Resource Description Framework). RDF дозволяє об'єднати інформацію з довільних джерел. Формат RDF найбільш корисний у забезпеченні спільного використання інформації, зміст якої може однаково інтерпретуватися різними програмними агентами. Специфіка моделі даних RDF полягає в тому, що ресурси й властивості ідентифікуються за допомогою глобальних ідентифікаторів (URI). RDF описує предметну область у термінах ресурсів, властивостей ресурсів і значень властивостей. RDF - дані можна розцінювати як сукупність тверджень - суб'єкт, предикат і об'єкт твердження, і представляти у вигляді спрямованого графа, утвореного такими твердженнями.

На рисунку 1 у термінах відповідних сутностей і зв'язків зображена загальна схема моделі RDF. Тут під властивістю (Property) варто розуміти якийсь аспект, характеристику, атрибут або відношення, що використовується для опису ресурсу. Кожна властивість має свій специфічний зміст, припустимі значення, тип ресурсів, до яких воно може бути застосовано, а також відносини з іншими властивостями. Для забезпечення унікальності імен властивості дотримуються концепції URI, тобто властивість стає потенційним об'єктом для опису за допомогою RDF окремо від ресурсу, що характеризується наявним значенням.

Базовий будівельний блок моделі даних RDF - твердження, що представляє собою трійку: ресурс, іменована властивість і його значення. У термінології RDF ці три частини твердження називаються відповідно: суб'єкт (subject), предикат (predicate) і об'єкт (object) [3]. Ресурсом у цьому

випадку називають усе, що описується засобами RDF. Це може бути звичайна Web-Сторінка або якась її частина, наприклад, окремий елемент HTML розмітки.

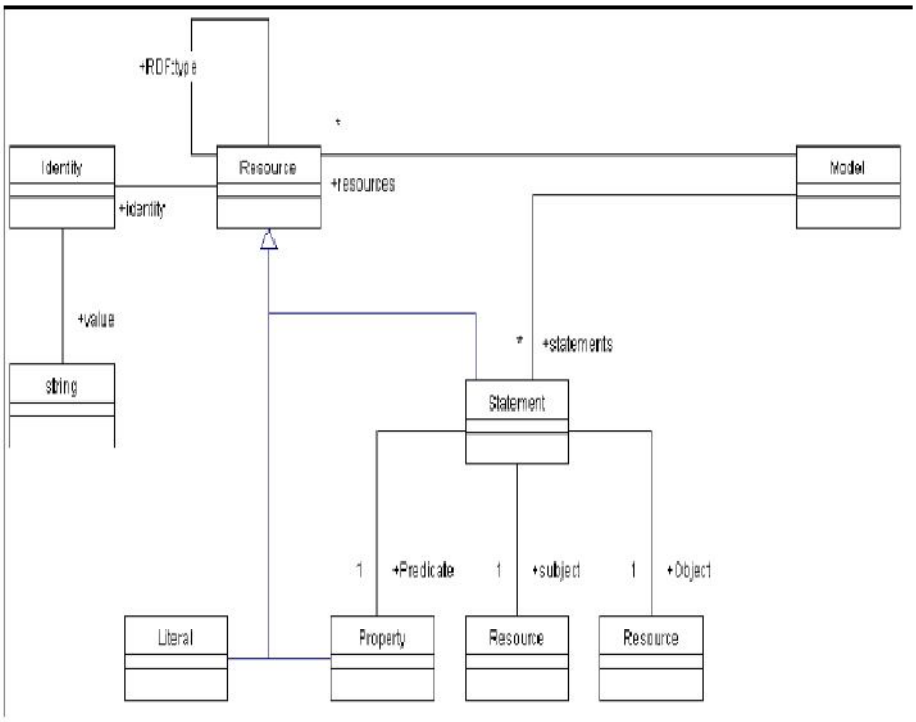

Рисунок 1 - Модель RDF

Висновок

У науковій роботі було здійснено аналіз існуючих підходів до розробки порталів знань на основі технологій Semantic Web. Встановлено, що для структуризації й класифікації інформаційних ресурсів необхідно використовувати підхід Topic Maps.

Здійснено реалізацію автоматичного рубрикування на основі семантичного аналізу вмісту інформаційних ресурсів. Даний підхід базується на певних способах подання знань про предметну область і текстової інформації. Він забезпечує значно кращу якість автоматичної класифікації ресурсів за рахунок того, що семантичний аналіз змісту ресурсу забезпечує більш достовірну оцінку приналежності ресурсу до тієї або іншої тематичної рубрики.

Список використаних джерел

1. https://www.w3.org/standards/semanticweb/
2. http://www.semanticweb.narod.ru/
3. http://www.alik.su/articles/semantic-web/semantic-web-2.pdf
4. "Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL" by Dean Allemang, May 20, 2011
5. "Learning SPARQL" by Bob DuCharme
6. "Programming the Semantic Web" by Toby Segaran and Colin Evans, Jul 24, 2009
7. "Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL" by Dean Allemang and James Hendler, May 9, 2008
8. http://semanticweb.org/wiki/Main_Page.html