Visit Notes Analysis Module

Implementer's Guide

Ryan Eshleman

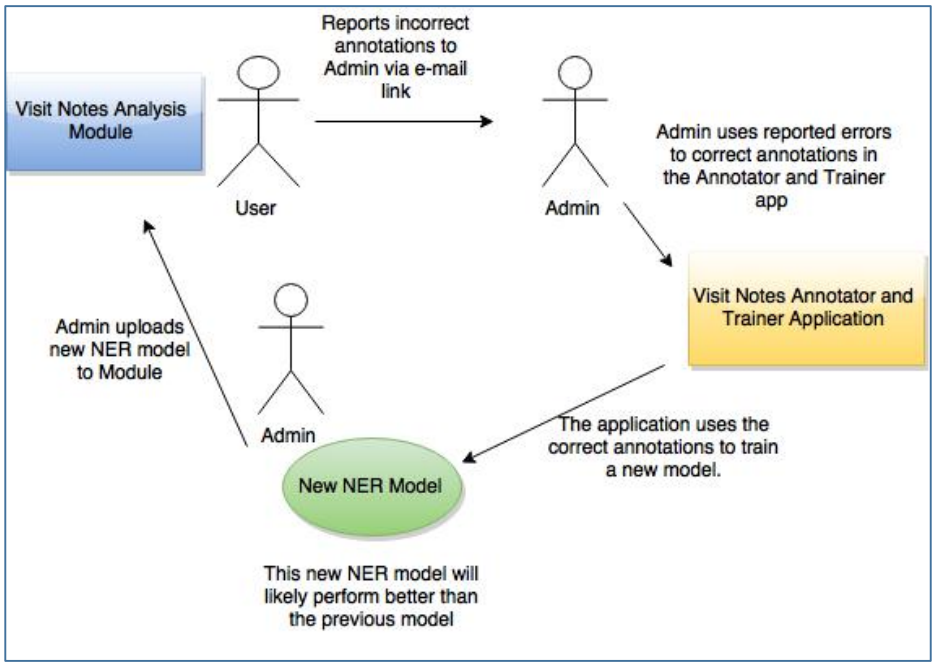
Advisors: Dr. Barry Levine
Dr. Hui Yang

Table of Contents:

Overview

This document provides implementation details for the Visit Notes Analysis Module as well as the companion web application, the Visit Note Annotator and Trainer. These two pieces work together to allow the optimal performance of the system. Figure 1 shows the circular workflow between the two applications. The Visit Note Annotator and Trainer is used to improve the performance of the Visit Notes Analysis module. However, it is worth noting that while these applications can improve the performance of the system, it is not 100% accurate. More details on this are provided in section 1.3.

Figure 1: Visit Notes Analysis and Visit Notes Annotator and Trainer application workflow.



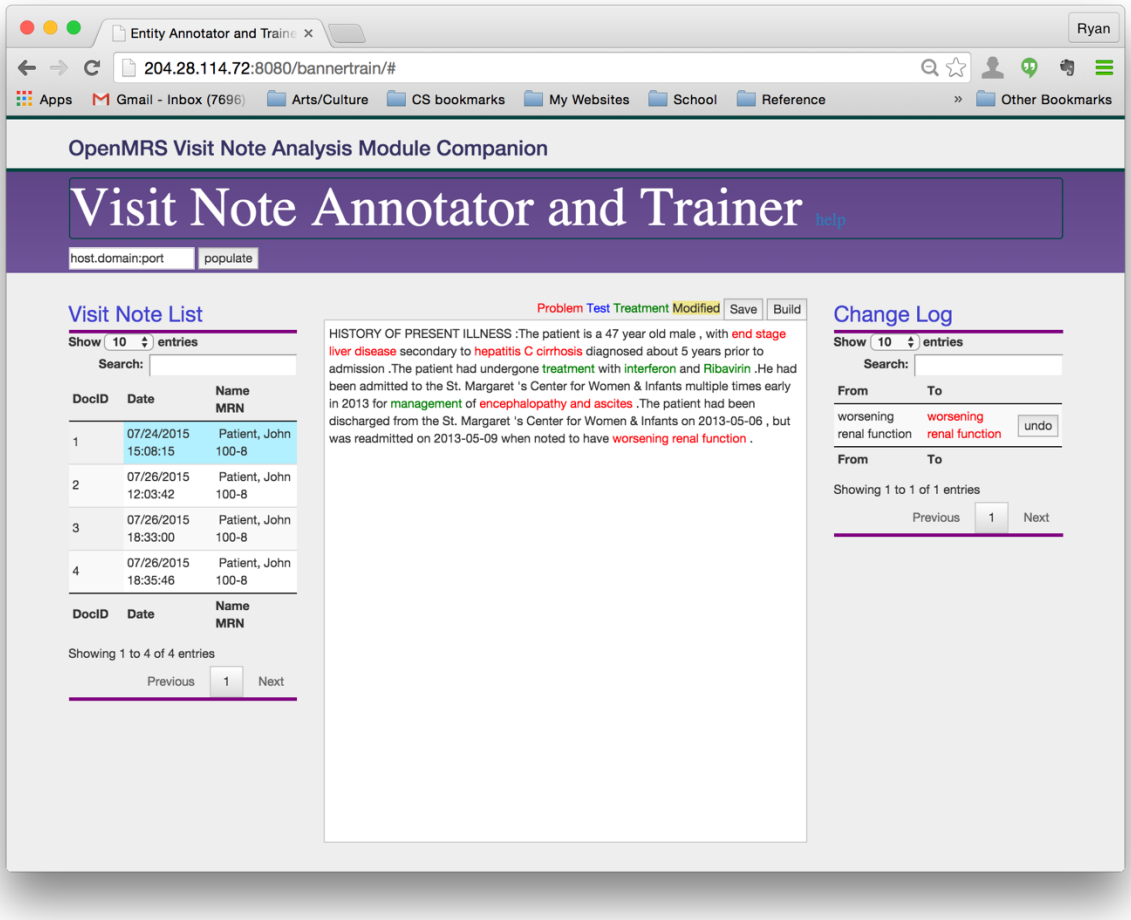
1.1 Visit notes Analysis Module Overview

The Visit Notes Analysis Module performs Named Entity Recognition (NER) on Visit Note text to identify where in the text a *problem, treatment or test* is mentioned. The underlying algorithm for identifying these entities has many moving parts, and this document will serve as a guide to configuring the NER functions for the best performance.

1.2 Visit Note Annotator and Trainer Overview

The Name Entity Recognition algorithm used to identify medical concepts in the text of Visit Notes relies in part on a machine learning algorithm called Conditional Random Fields, implemented in BANNER (http://banner.sourceforge.net/). Like many machine learning algorithms, in order to show strong performance, it must first be trained on sample data. In our case this sample data consists of Visit Notes where entities of the types "Problem," "Treatment," and "Test" have been annotated. The text area in Figure 2 shows an example of annotated data. The process of providing correctly annotated data to the algorithm for it to

Figure 2: The Visit Note Annotator and Trainer application



learn from is called training a model. The algorithm builds a model of what entities of the different types look like, and uses this model to identify entities in new text.

1.2.1 Why Train a new model?

The default model shipped with the Visit Notes Analysis module is trained on a public set of annotated clinical notes provided by I2B2 (i2b2.org). This model showed strong performance

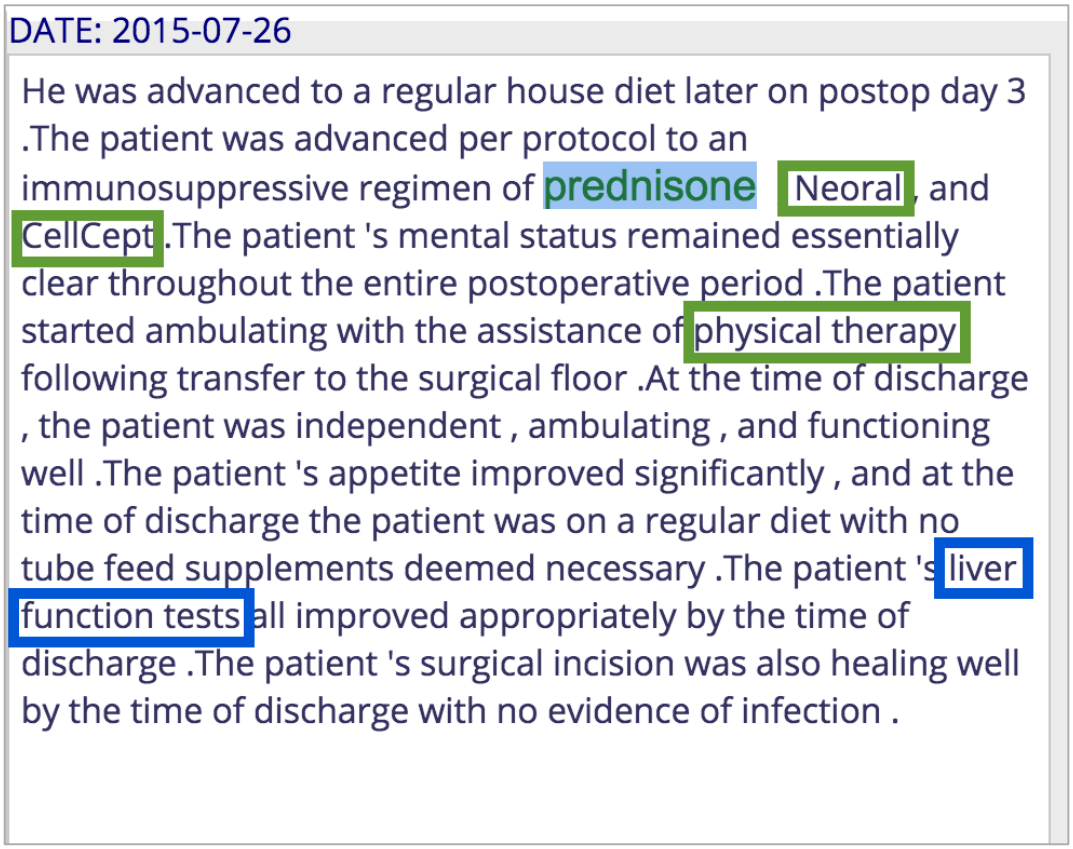
when used to tag documents in the I2B2 corpus, but textual and linguistic differences between I2B2 documents and Visit Notes encountered in the field might degrade the performance of the default model and cause it to overlook entities or mis-identify them.

We have provided the Visit Note Annotator and Training application to allow the OpenMRS administrators to use Visit Notes captured in their database to train a new model in the hopes that this model will improve performance of the system. Henceforth, when discussing the Trainer, the *user* will refer to the administrator responsible for training.

1.2.2 Example Use Case

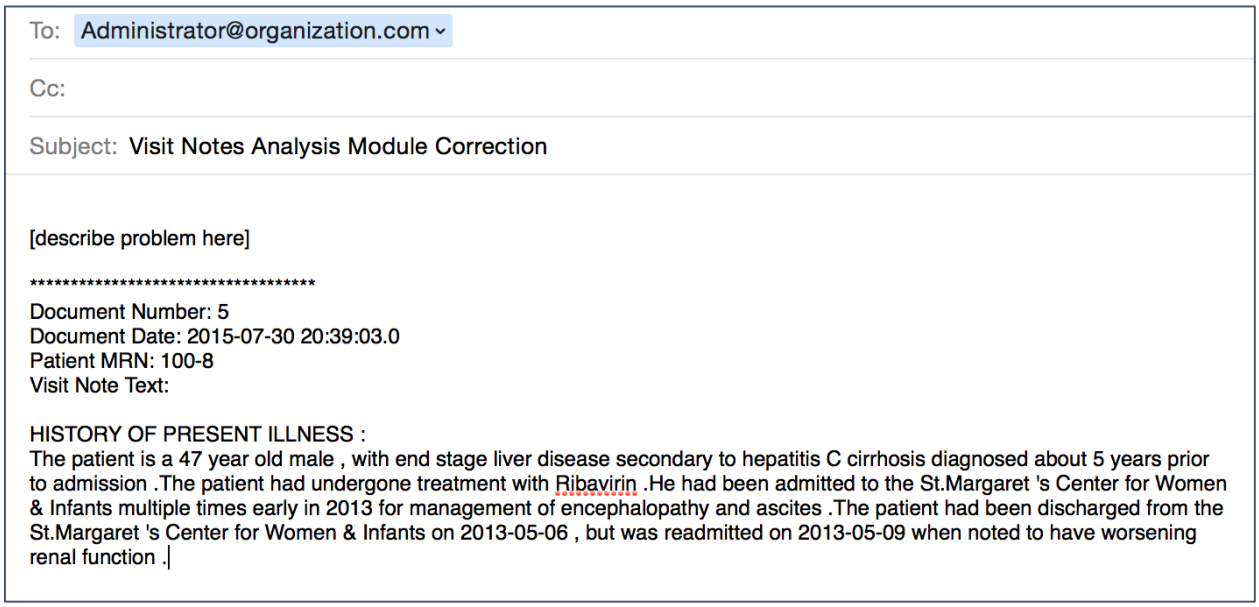
To illustrate how the OpenMRS module and Language Model Trainer application work together, consider the following use case. A user is examining the visit notes for patient John D. Patient. Figure 3 shows the annotations made to the Visit Note. An annotation is indicated by different text colors. Red for *problem*, green for *treatment* and blue for *test*. Here, "prednisone" is green, indicating it is a treatment. The user notices that one or more words in the visit note are not tagged correctly. "Prednisone" is correctly identified as a *treatment*, but "Neoral", and "CellCept" should have been identified as *treatments*, but were not, neither was "physical therapy". "liver function tests" should have been recognized as a *test*, but it was not.

Figure 3: Annotations by a model that performs poorly on this text. Many entities are not recognized.



A user (e.g. clinician using OpenMRS) can send an e-mail to the admin to report the incorrect annotations. An example tamplate e-mail is shown in figure 4. It provides the admin with the document number, the date and time the note was recorded, the patient Medical Record Number and the text of the Visit Note. This info will allow the admin to find and make corrections in the next steps.

Figure 4: Template email to admin to report errors.



The Admin can then navigate to the Visit Note Annotator and Trainer application, load the annotated visit notes, and correct the errors that were observed in the OpenMRS module. In figure 5, the documents are loaded and we can see that the mistakes match those found in the OpenMRS module. Figure 6 shows the corrections to the annotations made by the admin. These corrections are reflected both on the document rendering as well as the Change Log.

Figure5: Poorly annotated Visit Note loaded into application.

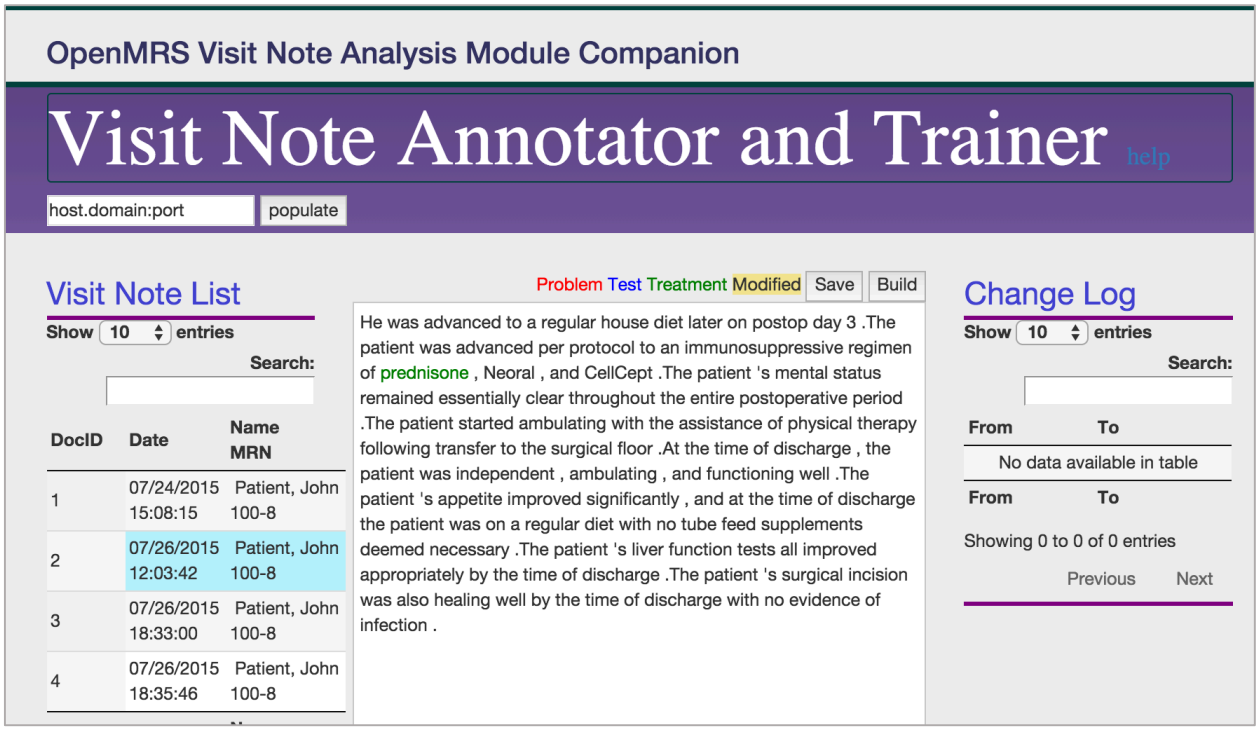
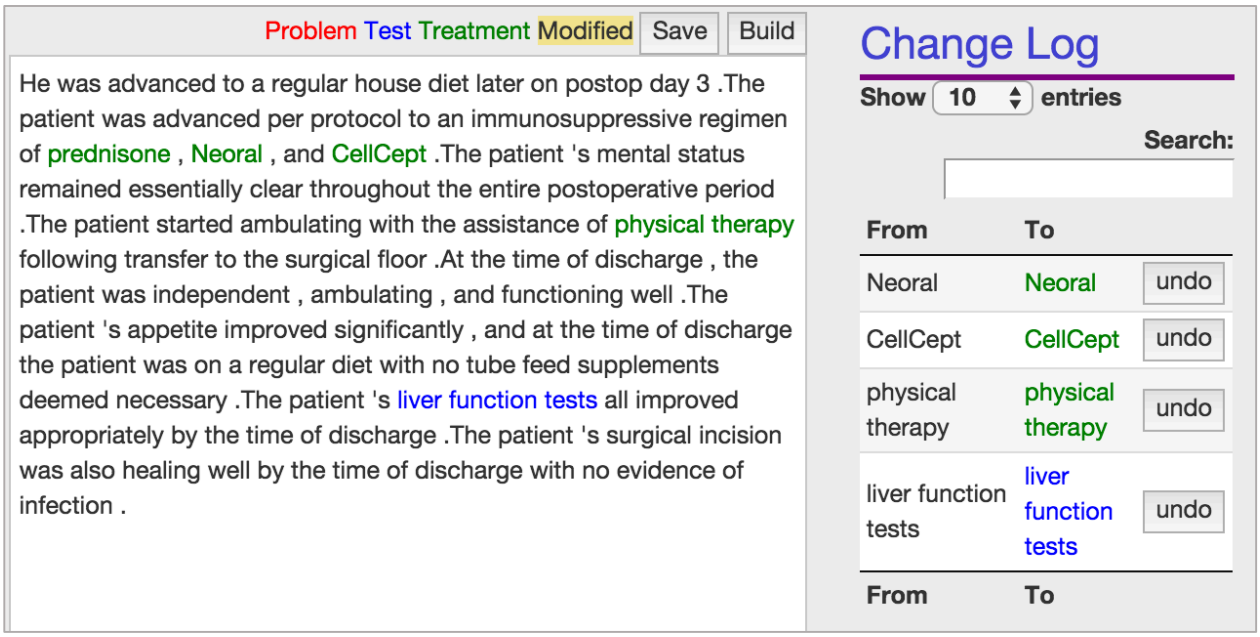


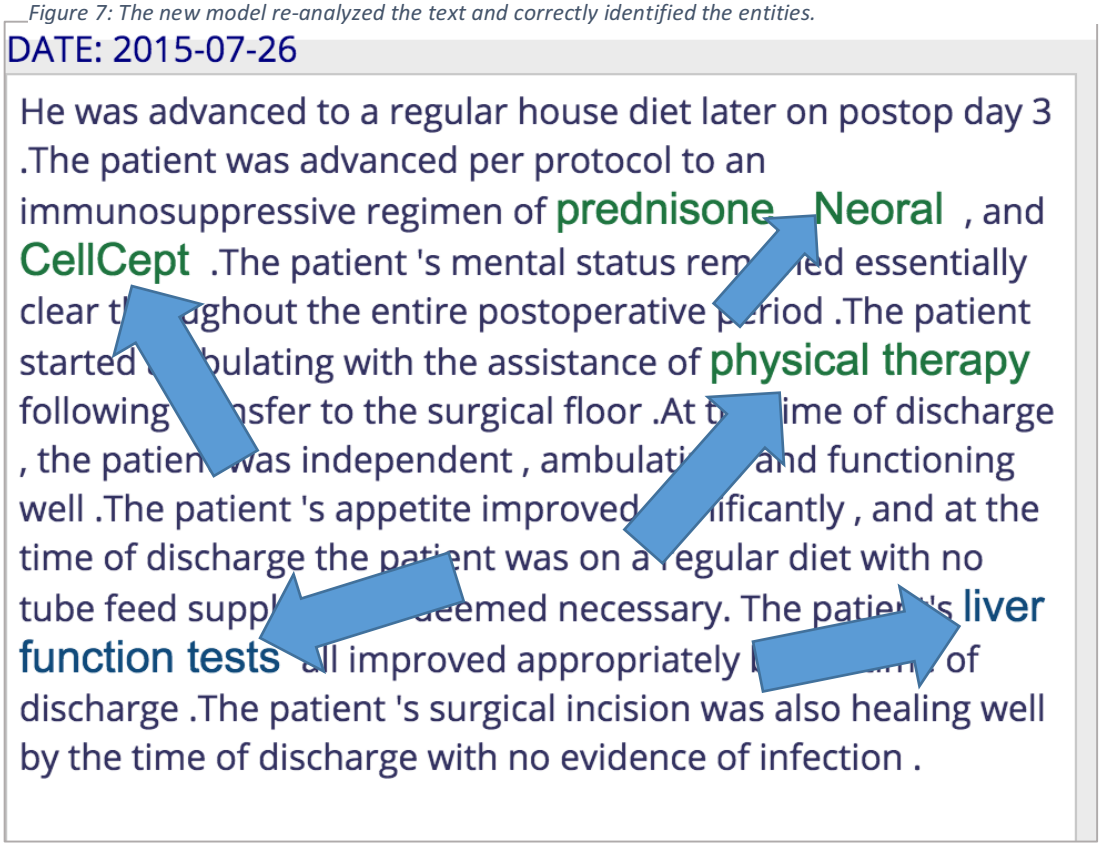
Figure 6: Admin corrects the annotations



When the admin is satisfied with the document annotations, the admin selects the "Build" button and the text and annotations are used to train a new BANNER Language Model. The admin is able to download this model from the training application and upload it to the OpenMRS implementation via a configurations page.

This new model is used to re-analyze the text and it is able to successfully find the correct entities in the text. Figure 7 shows the results of the new analysis.
In this way, the OpenMRS can improve their language model as they encounter errors.

Figure 7: The new model re-analyzed the text and correctly identified the entities.



1.3 Limitations

During the development of this module, we performed evaluations of several NER algorithms in order to choose the best one to use here. During these evaluations it became clear that the current state of NER is not perfect. This module uses the BANNER NER system and our measurements showed that of the entities it finds, about 80% are correct. Also, it tends to find about 70% of the entities in the text. While this performance is good, it indicates that the results of this module should not be relied upon for perfect accuracy, and should be used as a tool to assist in analyzing Visit Notes.

Installation

2.1 Visit Notes Analysis Module Installation

Requirements:

Jdk 1.7
At least 2GB available RAM
Requires OpenMRS versions 2.x, core 1.11.x
[Support for 1.9x is not available yet.]

Pre-Installation – Server Configuration (Tomcat)

Ensure server JVM has at least 2GB of heap space available

1. Include this line to $CATALINA_HOME/bin/setenv.sh:
   ```
   CATALINA_OPTS="-Xms512M -Xmx2048M"
   ```
   On a standard tomcat installation on Ubuntu, this is in `/usr/share/tomcat6/bin`

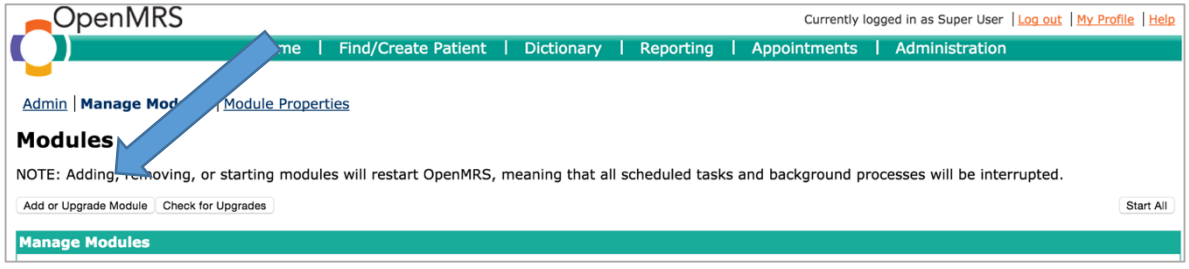
Installation

Installation of the module follows the standard OpenMRS module installation procedures.
NOTE: installing this module requires you to restart your OpenMRS installation

1. Navigate to OpenMRS Administration Page
2. Select the “Manage Modules” Link
3. Upload the .omod file through the “add or upgrade module” button.
   Note: the .omod file is rather large ~48MB so the upload may take some time

4. Restart your OpenMRS installation.

Figure 8: Upload the module via OpenMRS’s Manage Modules page



2.2 Visit Notes Annotator and Trainer installation

This application has been tested in the following environment:

Java Servlet Container:	Apache Tomcat 6 servlet container
JVM:	Java 1.7
OS:	Ubuntu Linux 14.04

For current Tomcat installation instructions, please refer to one of the many Tomcat installation guides available on the web. For example:

https://help.ubuntu.com/lts/serverguide/tomcat.html

Note: be sure to install the Tomcat administration pages webapps.

Installing the application via Tomcat Application Manager

1. Navigate and login to the Tomcat Application Manager, this is often at [server-URL]:[port]/manager/html.

Figure 9: Tomcat Application Manager page

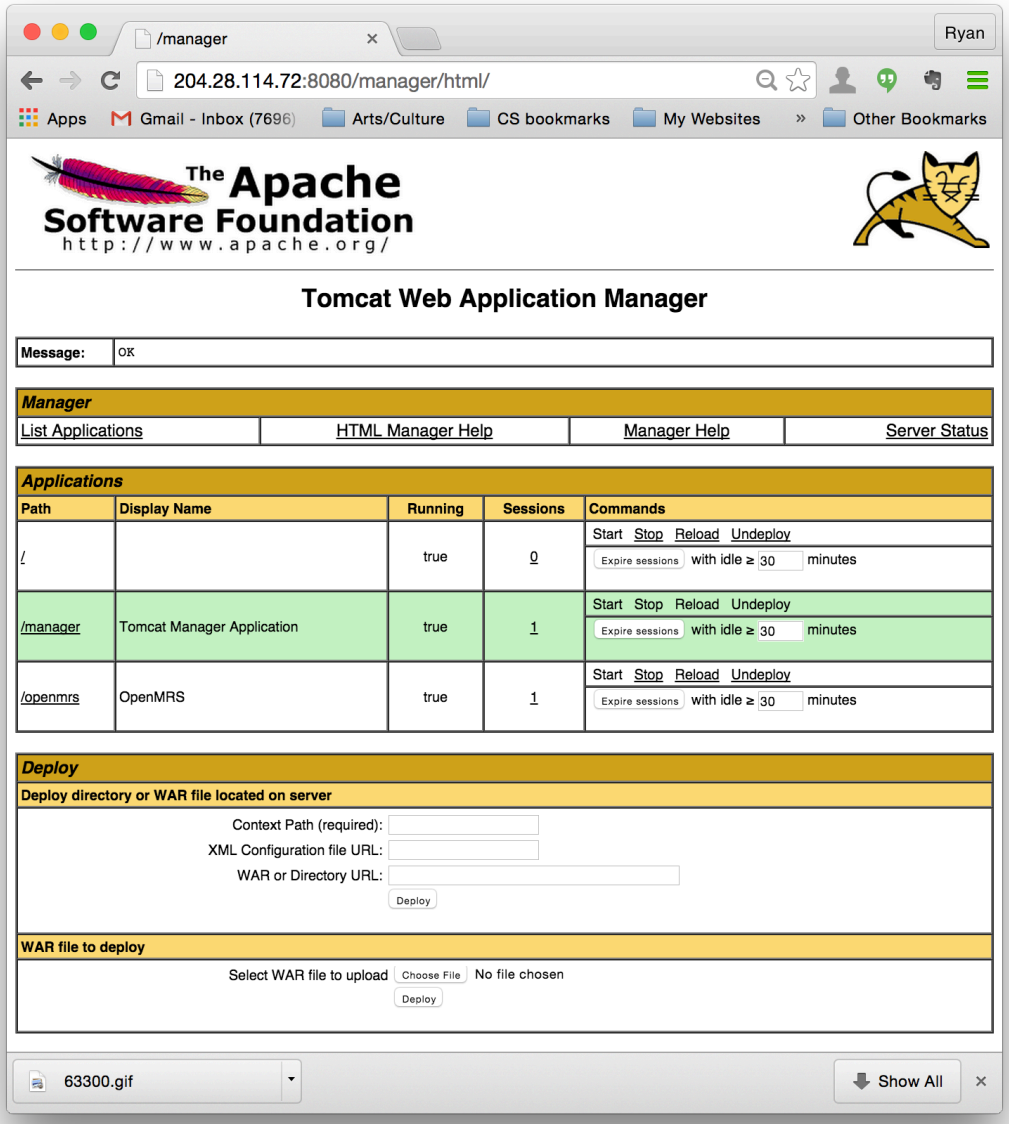
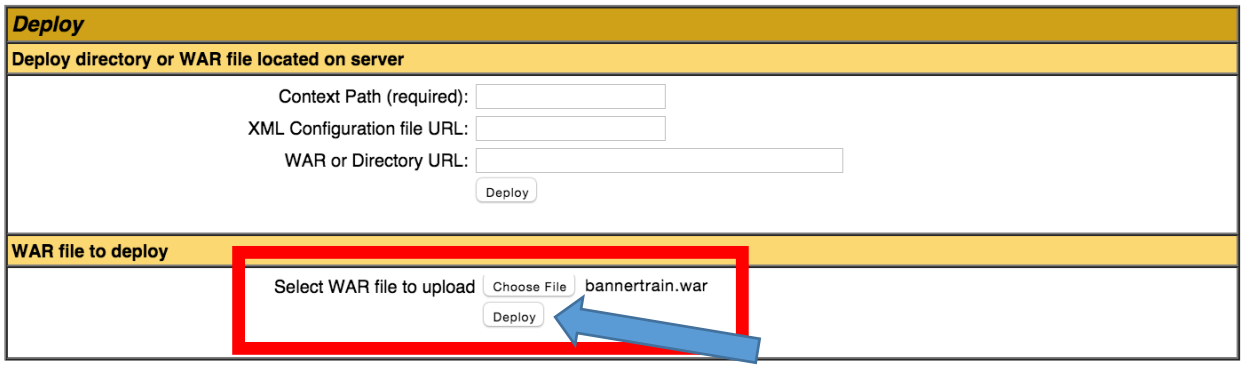


Figure 10: upload the .war file



2. upload the bannertrain.war file via the "WAR file to deploy" pane
3. Click "Deploy"
4. After the files finishes uploading to the server (may take several minutes, depending on connection). The application will be visible in the "Applications" pane, figure 11.

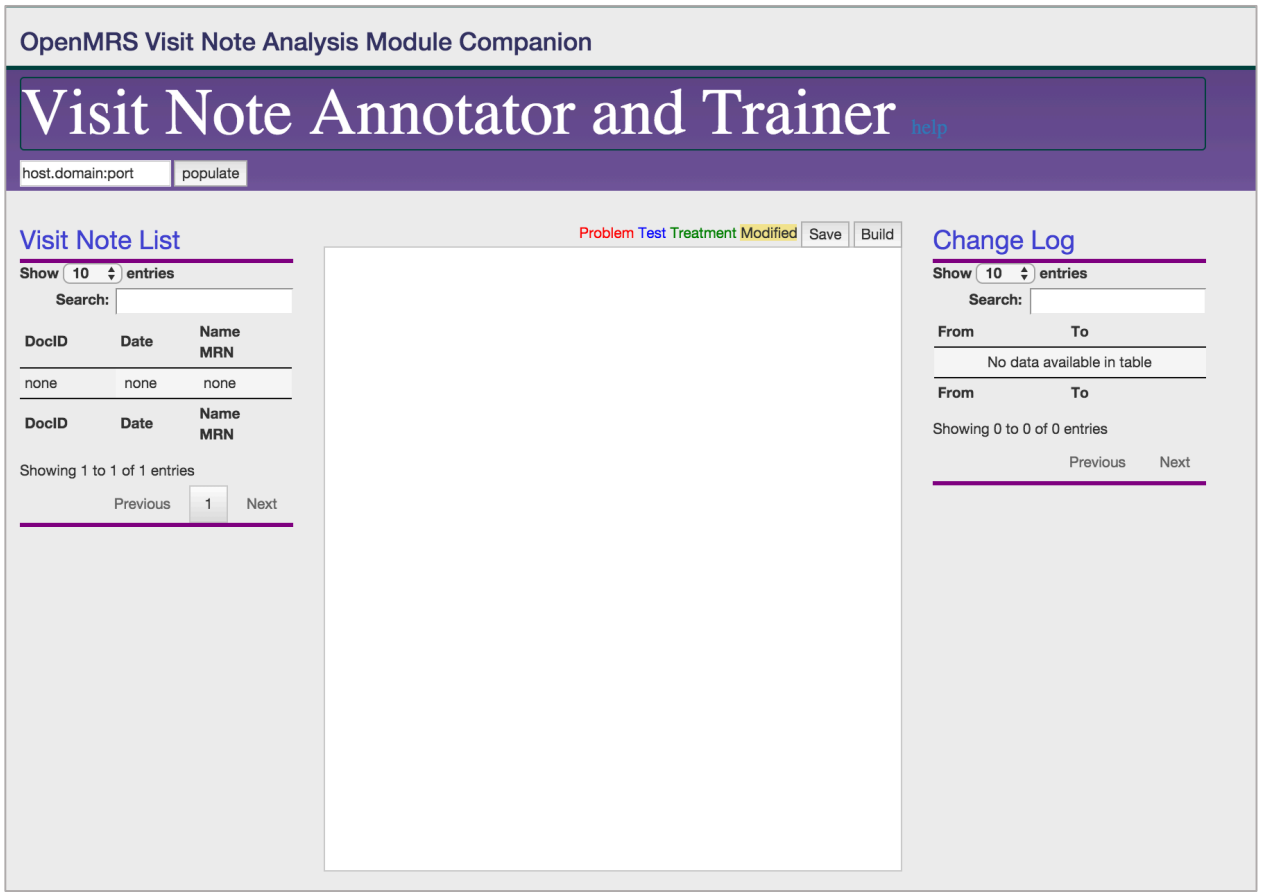
Figure 11: The application is running and accessible at the indicated path

Applications

Path	Display Name	Running	Sessions	Commands
/		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/bannertrain	bannertrain	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/openmrs	OpenMRS	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

5. Clicking through the link will bring you to the deployed web app, figure 12.

Figure 12: Visit Note Annotator and Trainer application with no Visit Notes loaded



Module Configurations and Maintenance

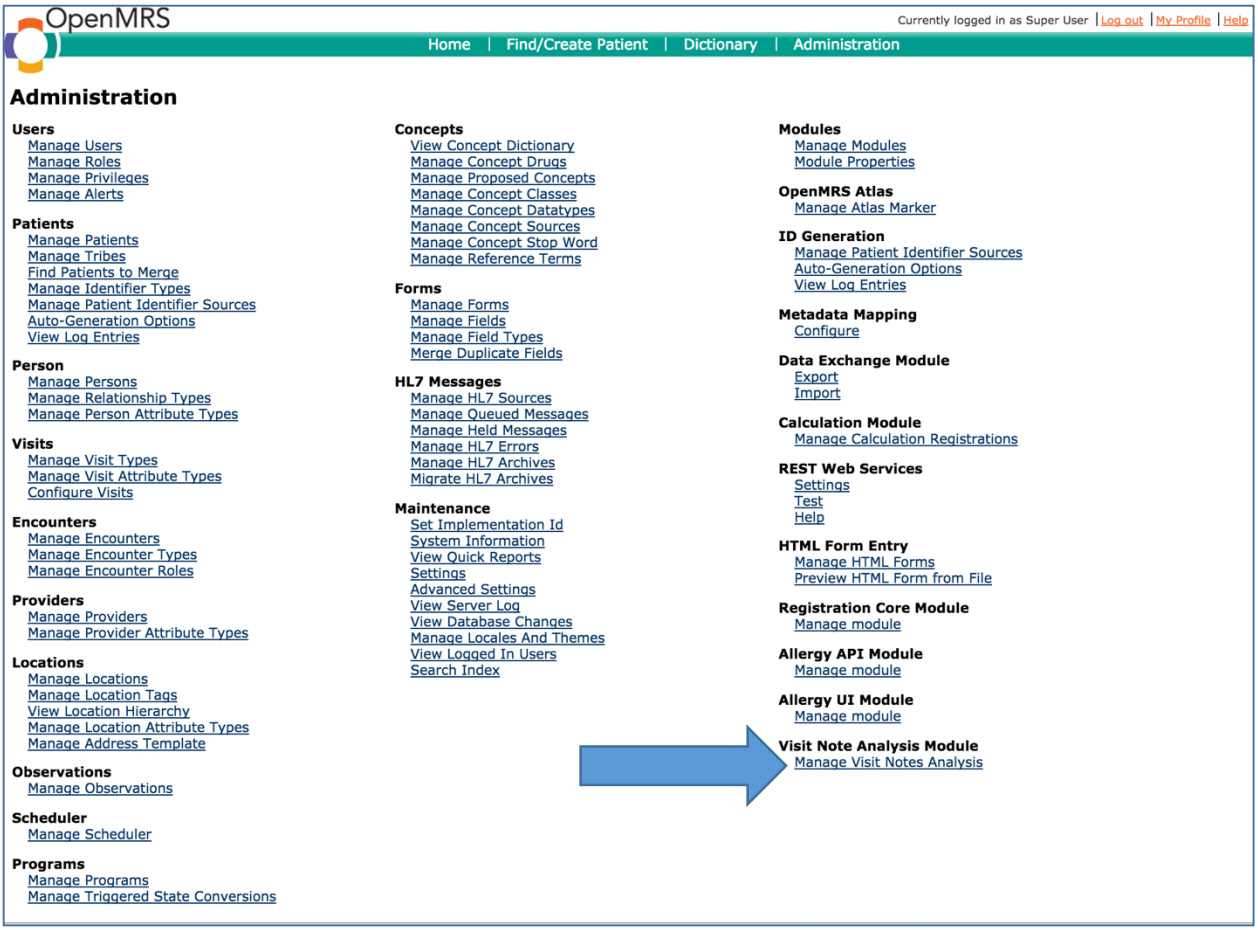
3.1 Defining Concept Class – Entity Mappings

The first step of the Named Entity Recognition algorithm uses the Concept Dictionary to perform string matching against concepts in the dictionary. Because our module tags entities into three distinct entity types but an OpenMRS implementation often has many more concept classes, we allow the user to provide a many-to-one mapping from OpenMRS concept classes to entity type.

To set this mapping:

1. Navigate to the “Manage Module” page from the OpenMRS Administration page

Figure 13: OpenMRS Administration page



2. Locate the input fields labeled “Problems”, “Treatments”, and “Tests.” The values should be a comma separated list of the Concept Classes that match to each Entity class. A list of concept classes can be found at the “Manage Concept Classes” link from the Administration page. The “Manage Tagger” sections maintains configurations related to the NER tagger.

Figure 14: Setting Concept Class – Entity Type mapping

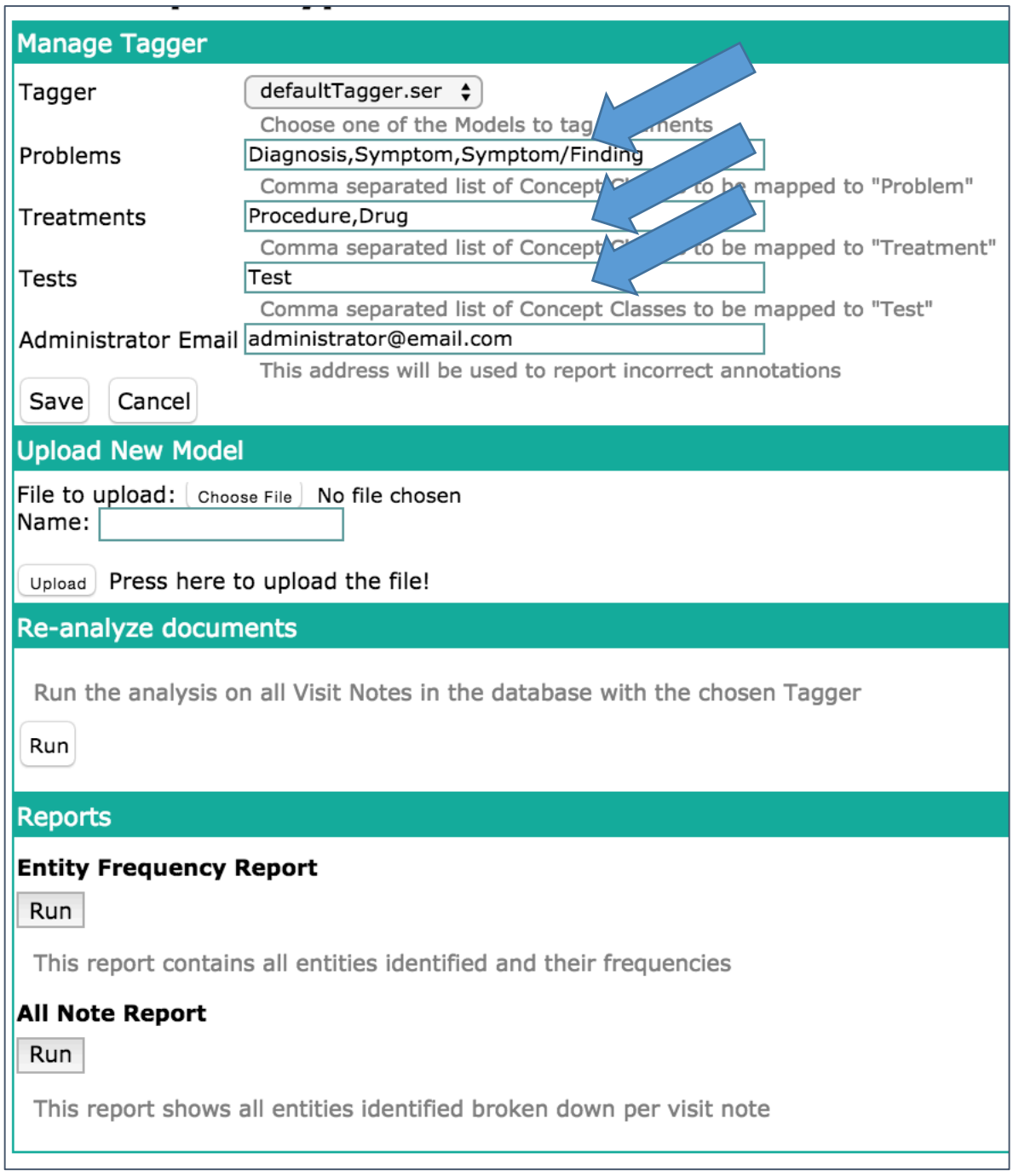


Figure 15 shows how to navigate to the Concept Class Manager from the Administration page and figure 16 shows the list of Concept Classes:

Figure 15: Administration page – finding Concept Classes page

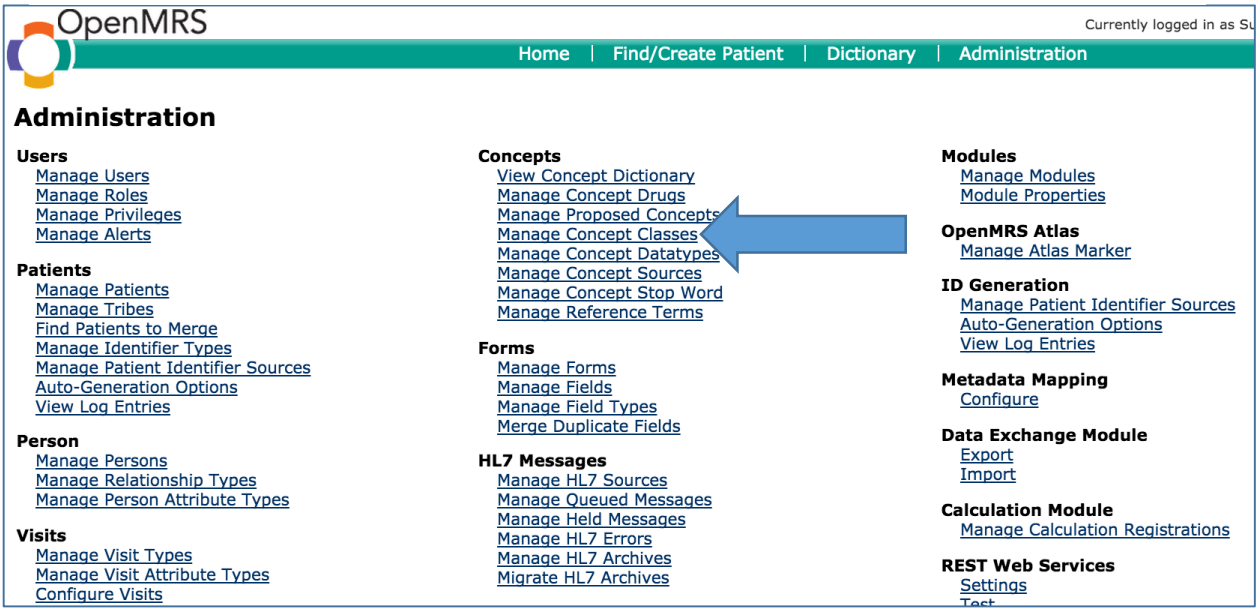
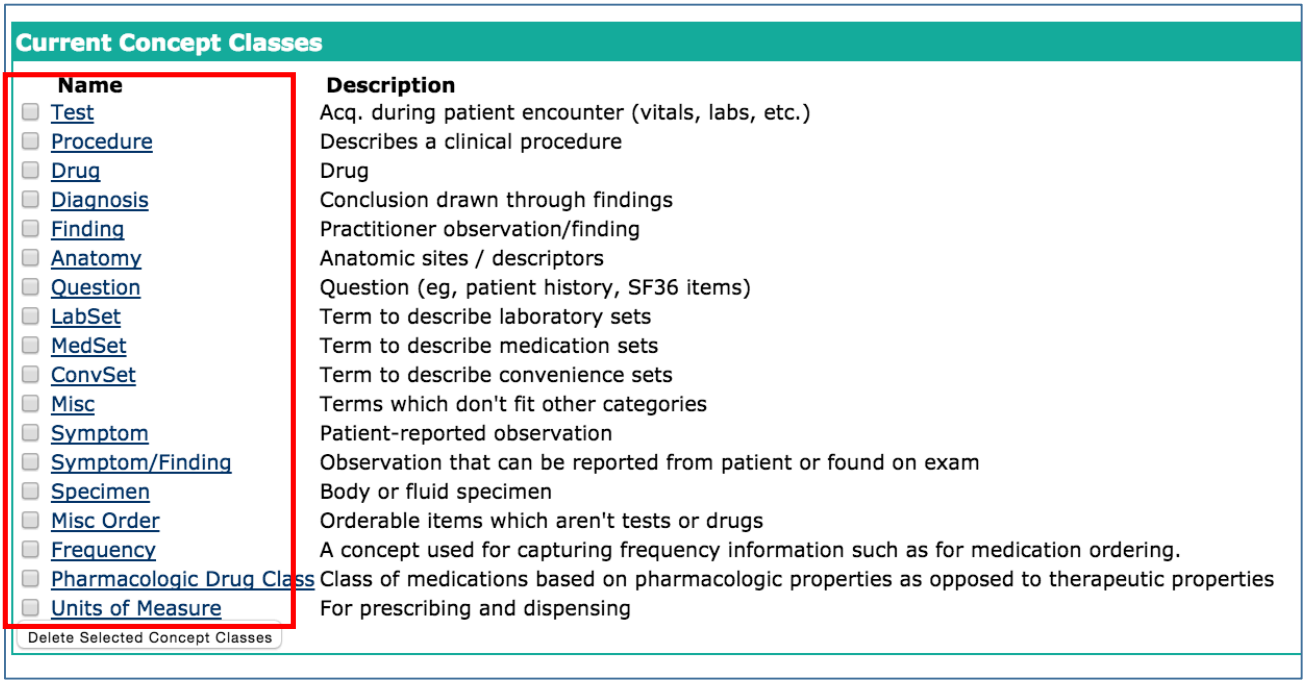


Figure 16: Concept Classes shown inside red box



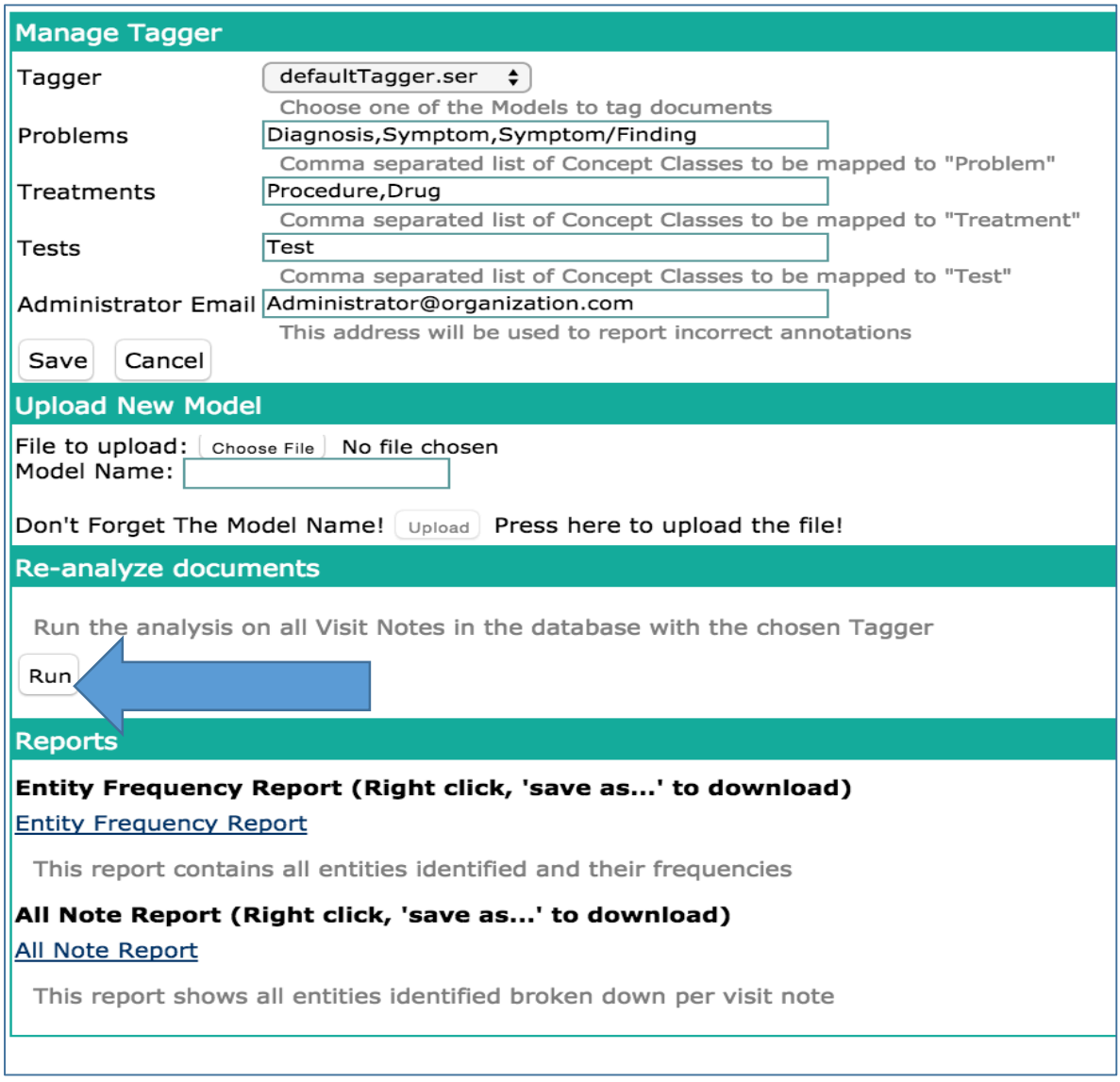
3.2 Analyzing and Re-analyzing Visit Notes

When the module is first installed, there might be a corpus of visit notes existing in the current database. The module does not automatically run its analysis on these documents. Once installed, it will run the analysis on all Visit Notes that are subsequently submitted. Our module provides the ability to retroactively run the text analysis on Visit Notes currently in the Database.

To analyze all Visit Notes in the database:

1. Navigate to the "Manage Visit Notes Analysis" page from the OpenMRS Administration.

Figure 17: Re-running Visit Note analysis



2. Select "Run" to run the text analysis on all Visit Notes in the database, figure 17.

NOTE: As there may be a very large number of Visit Notes in the database, this job may take a while and consume considerable resources. We recommend running this job during low demand, low traffic periods.

3. Results from this analysis will now be visible through the "Visit Notes Analysis" action on the patient dashboard, figures 18 and 19.

Figure 18: Patient Dashboard shows how to find the Visit Notes Analysis module

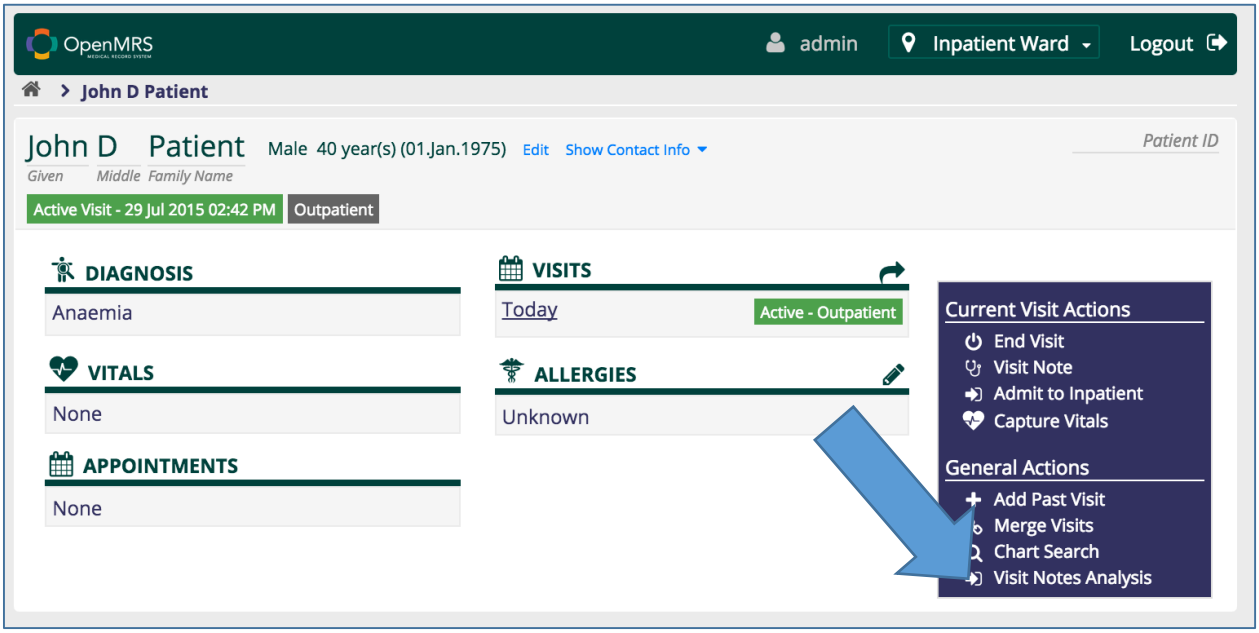
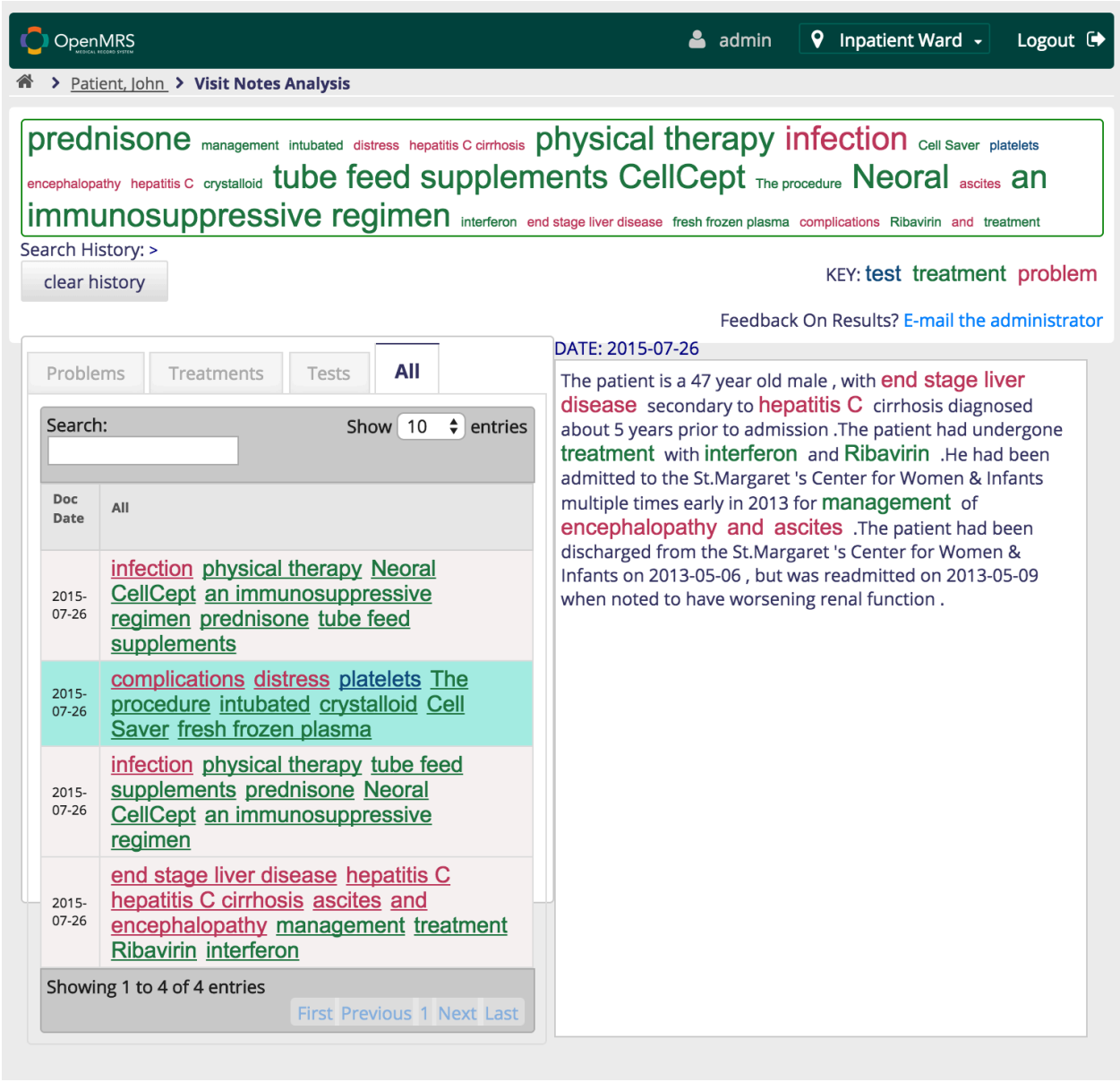


Figure 19: Results of NER analysis visible from the Visit Notes Analysis page



3.3 Adding a New Model

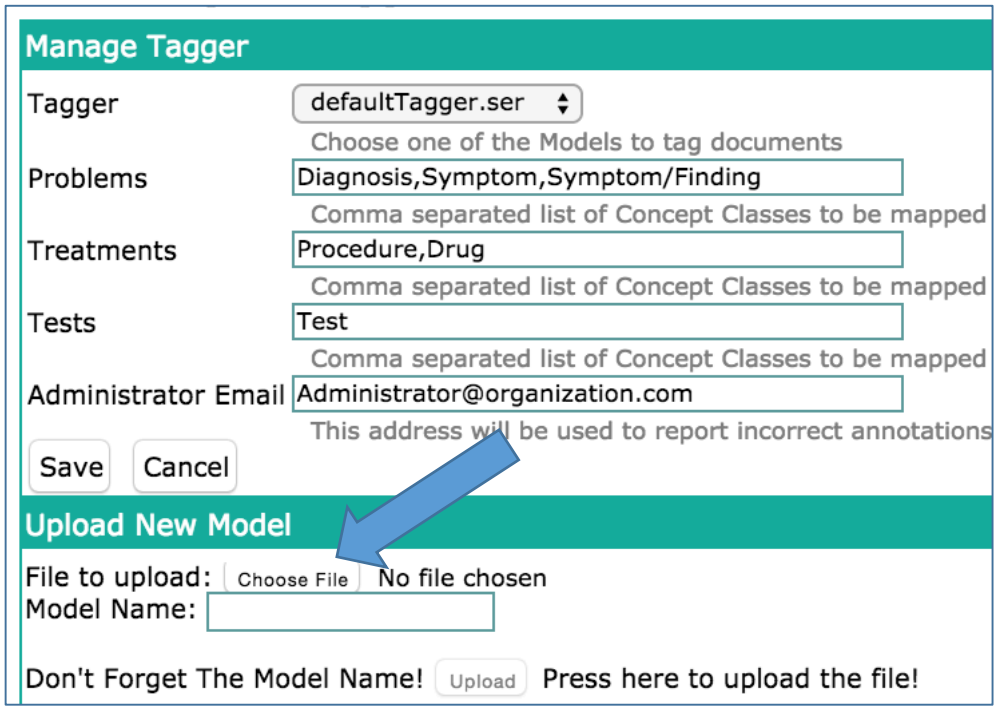
The companion application "Visit Note Annotator and Trainer" allows the user to build a new model to improve the module's ability to recognize medical entities. See section 6 of this guide for instructions on building a new model. Once the Trainer has built the model file, follow these steps to upload the model into the module.

Uploading a new Model:

1. Navigate to the “manage Visit Notes Analysis” page from the OpenMRS Administration page.

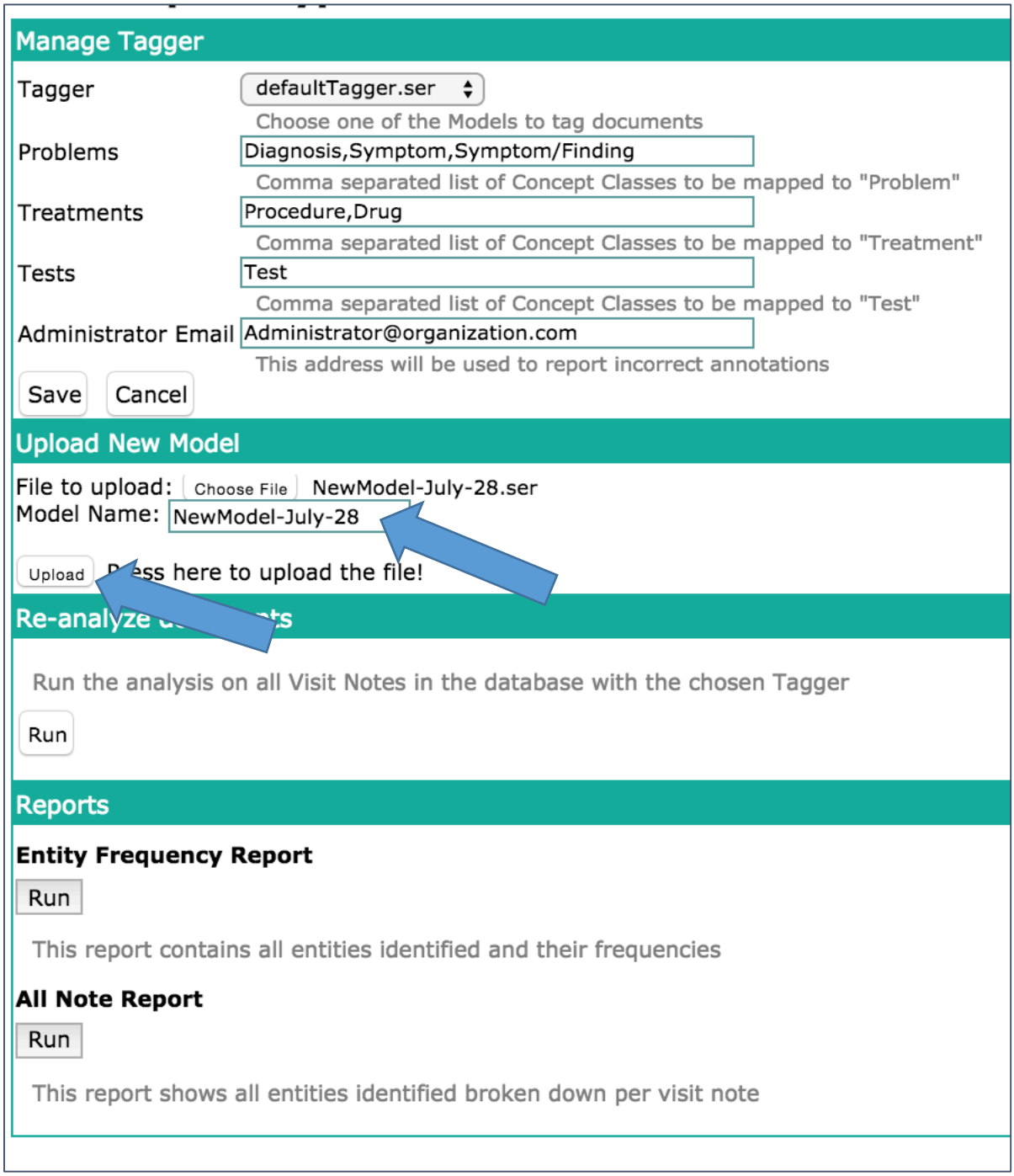
2. Select “Choose File” and find the model file on your local file system, figure 20.

Figure 20: Uploading a new model



3. Input a name for your model, figure 21.
4. Select “upload,” figure 21.
5. The new model is now uploaded into the module and can be chosen for tagging in the future

Figure 21: Name and save the model



3.4 Choosing a Model

This module is distributed with a default tagging model and allows for new models to be uploaded. To configure which tagger your module will use, follow these instructions

Choosing a Tagger:

1. Navigate to the Manage Visit Notes Analysis page.

Figure 22: choosing a tagger from the dropdown

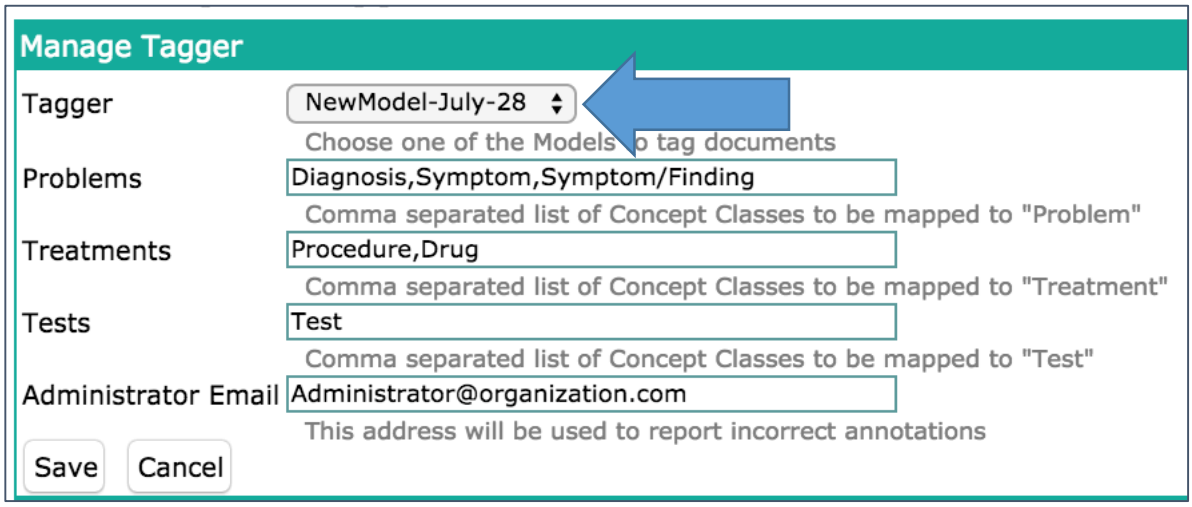


Figure 23: Select your tagger

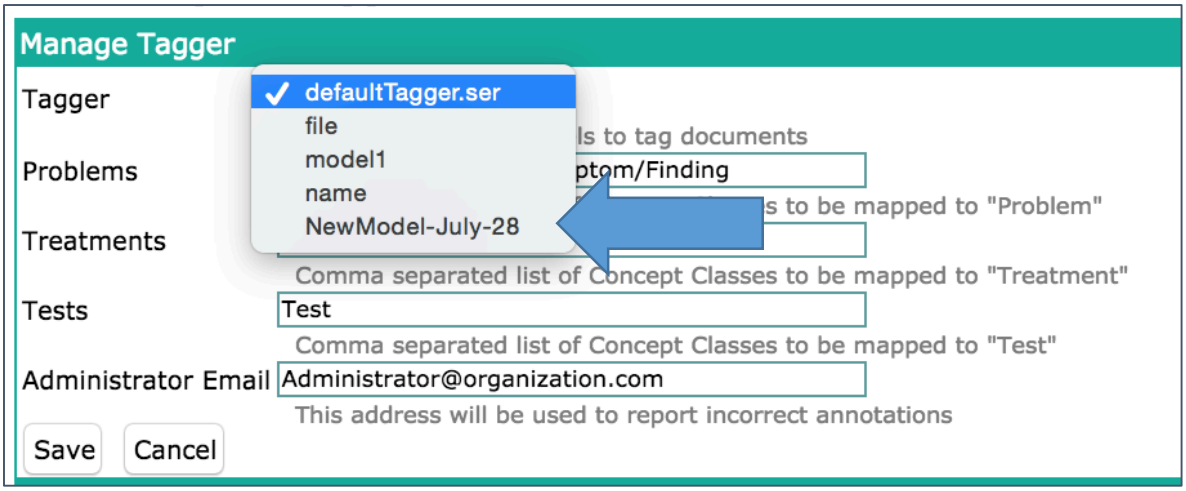
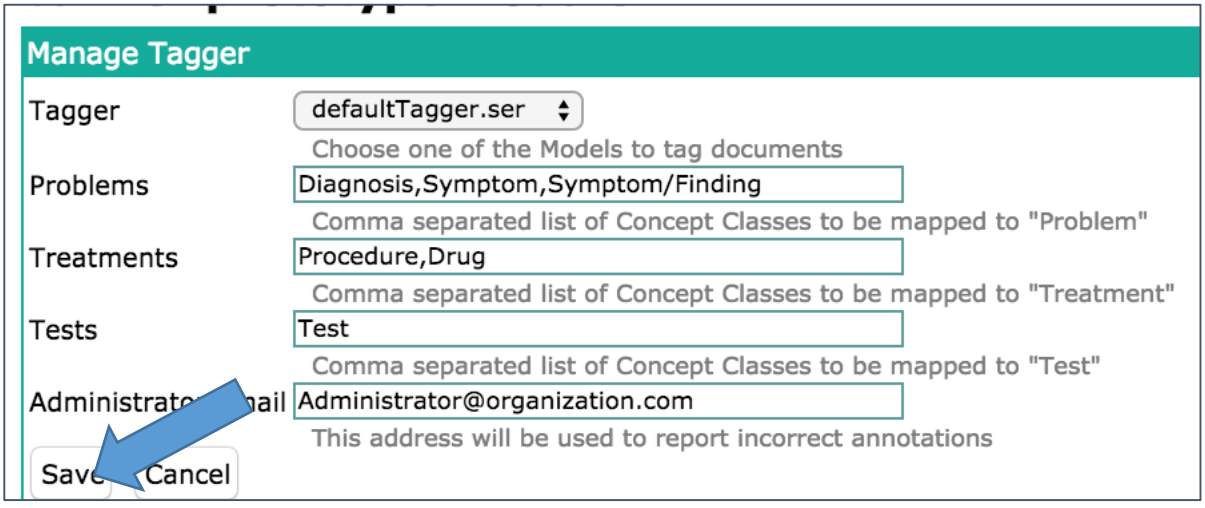


Figure 24: Save this new setting



2. select the dropdown list under “Manage Tagger,” figures 22 and 23.
3. Save this new setting by clicking the “save” button, figure 24.
4. NewModel-July-28 will now be used to tag new Visit Notes.
5. Optionally, the admin can now follow the steps in section 3.2 to re-analyze the existing corpus of visit notes using the new model. All old models are saved (see the dropdown menu in figure 23) and can be chosen for use by following the steps in this section.

3.5 Running Reports

The Visit Notes Analysis module allows you to run two reports at this time. The Entity Frequency Report and the All Notes Report.

The Entity Frequency Report generates a csv file with all entities and their entity type along with the number of occurrences of that entity in the entire visit note corpus. Output is in the format:

```
Entity,type,frequency.
```

For example:

```
Nausea,problem,5
cd4,test,4
```

The All Notes Report generates a csv that shows all notes analyzed in the system and the entities in them. This note shows one note per line in the format:

```
patientID,DocID,date,entities&&types
```

entities&&types is a vertical bar separated list of all entities found in the note DocID is a unique document identifier. An example line in this report:

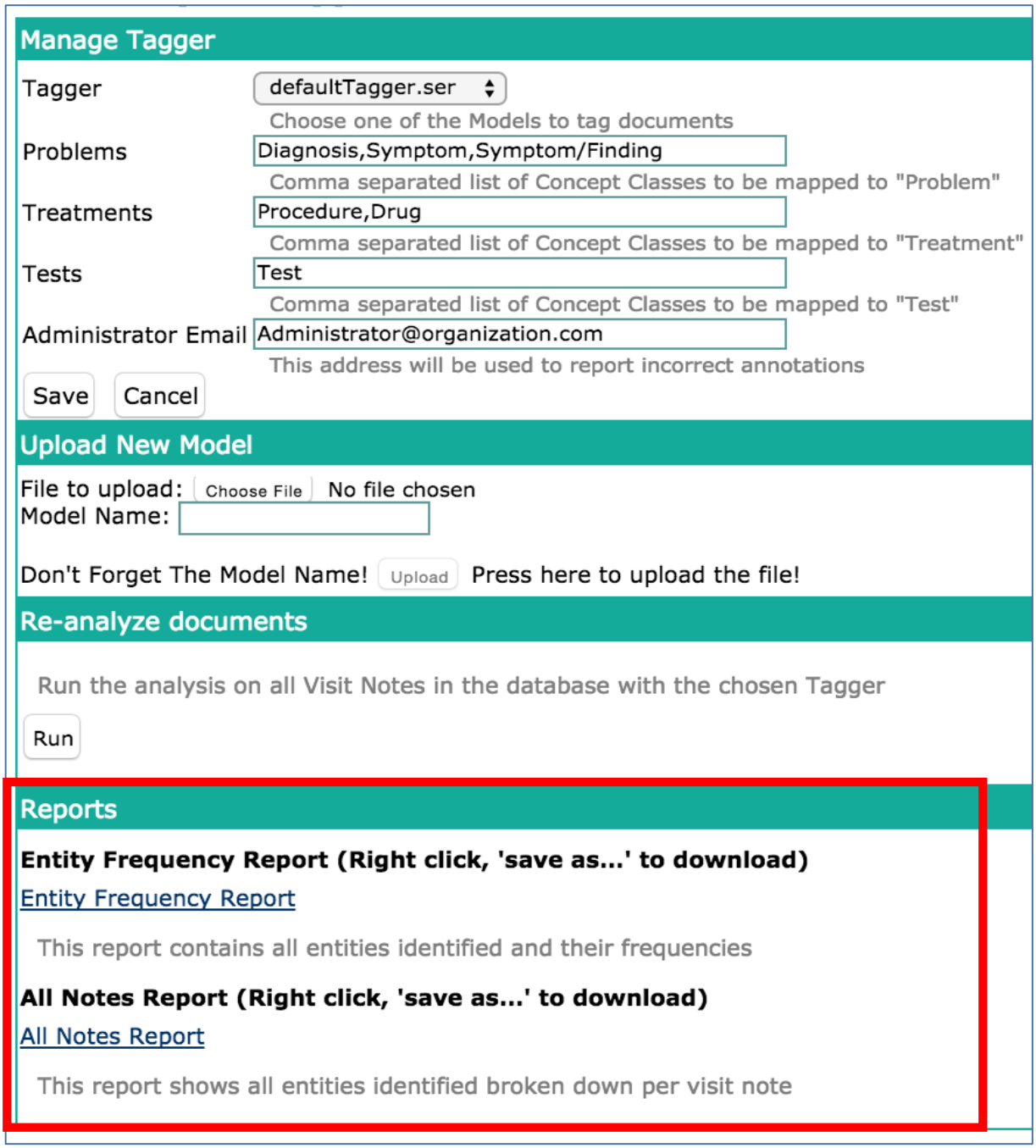
```
3,1,2015-07-29 14:42:48.0,nausea&&problem|vomiting&&problem
```

To run the reports, follow these instructions;

Running Reports:

1. Navigate to the Manage Visit Note Analysis page
2. Find the section labeled “Reports,” figure 25.
3. This section contains links for the Entity Frequency Report and the All Notes Report

Figure 25: Running Reports



4. Clicking on the link will render the report in the browser, figures 26 and 27.

Figure 26: Selecting a Report

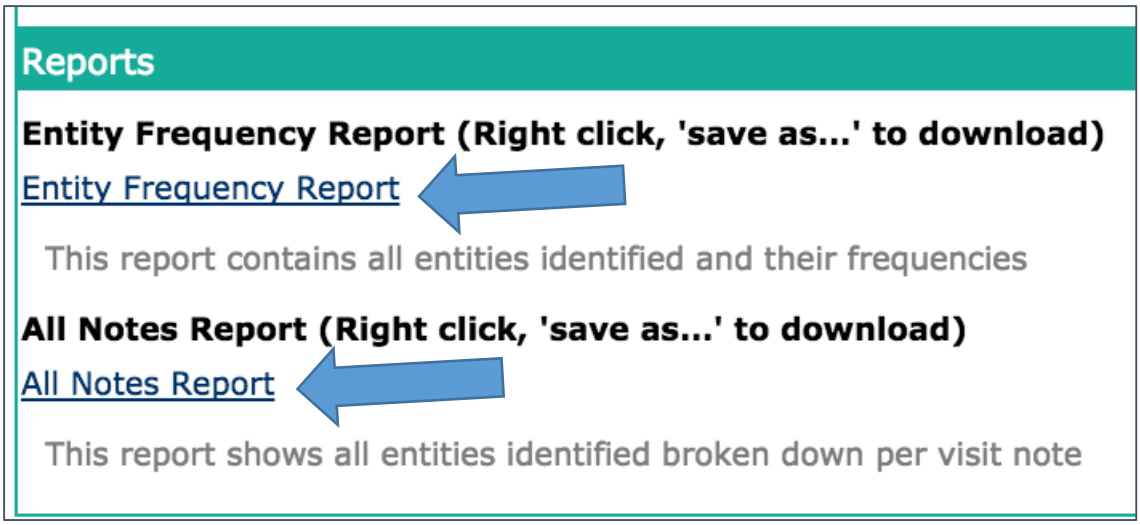


Figure 27: Entity Frequency Report

```
Entity,Type,Frequency
the blue elephant aspergum,problem,1
aspirin,treatment,1
cat scratch fever,problem,1
a burning pain,problem,1
left upper quadrant pain,problem,1
resection,treatment,1
blood transfusions,treatment,1
fever,problem,1
the pain,problem,1
heterosexual transmission,problem,1
similar pain,problem,1
DDI,treatment,1
HIV positive,problem,2
aspergum,treatment,2
any anti retroviral therapy,treatment,1
HIV,problem,2
an abscess in the left lower extremity,problem,1
vomiting,problem,3
presumed,problem,1
nausea,problem,2
pancytopenia,problem,1
```

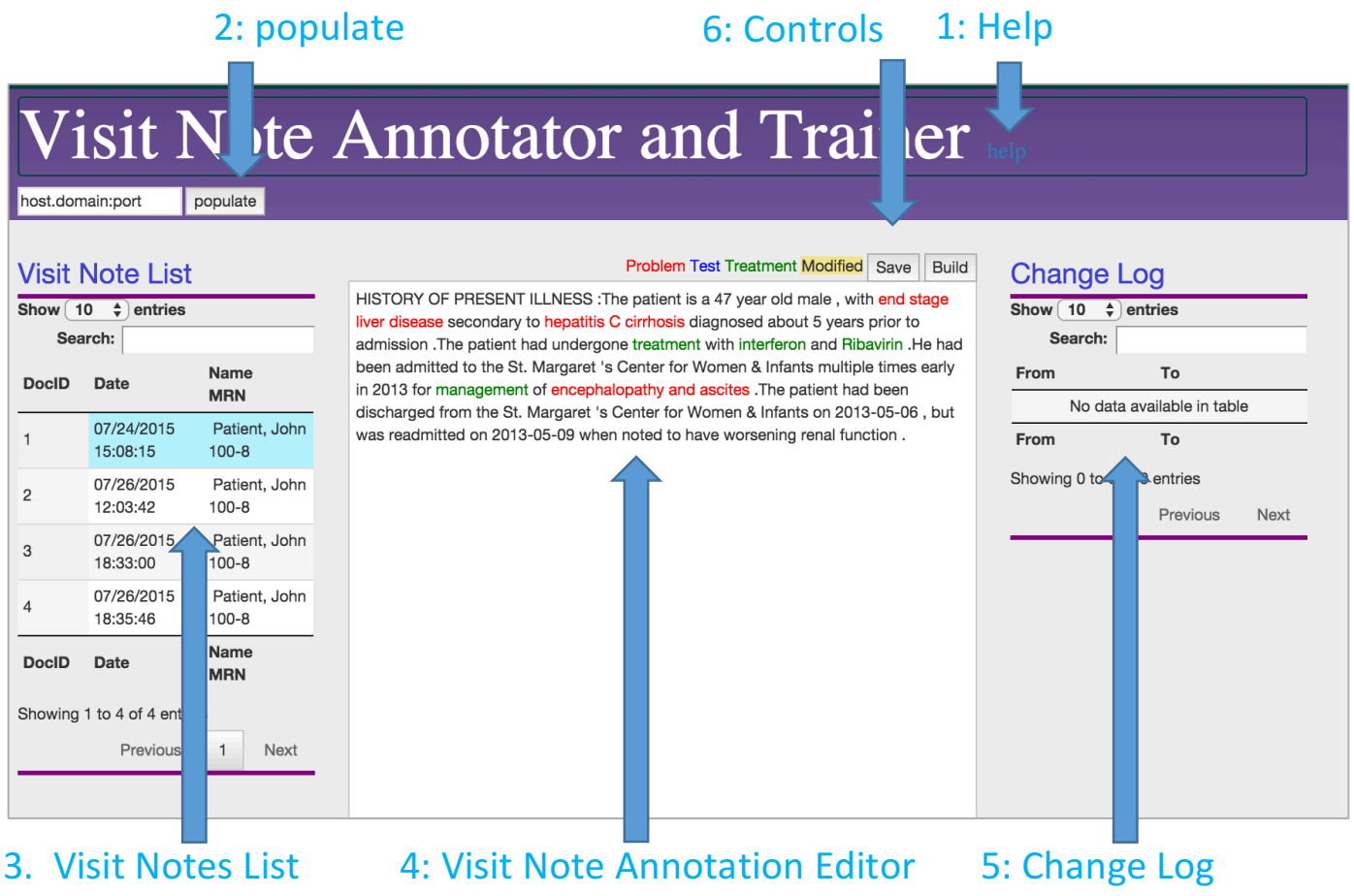
5. Right click on the link and select “Save as…” to download the report to your local machine.

Visit Note Annotator User Interface

The UI consists of 6 major components. They are labeled in figure 28, followed by a brief description. Usage details are provided in Section 5. The components are as follows:

1. Help
2. Populate
3. Visit Notes List
4. Visit Note Annotation Editor
5. Change Log
6. Controls

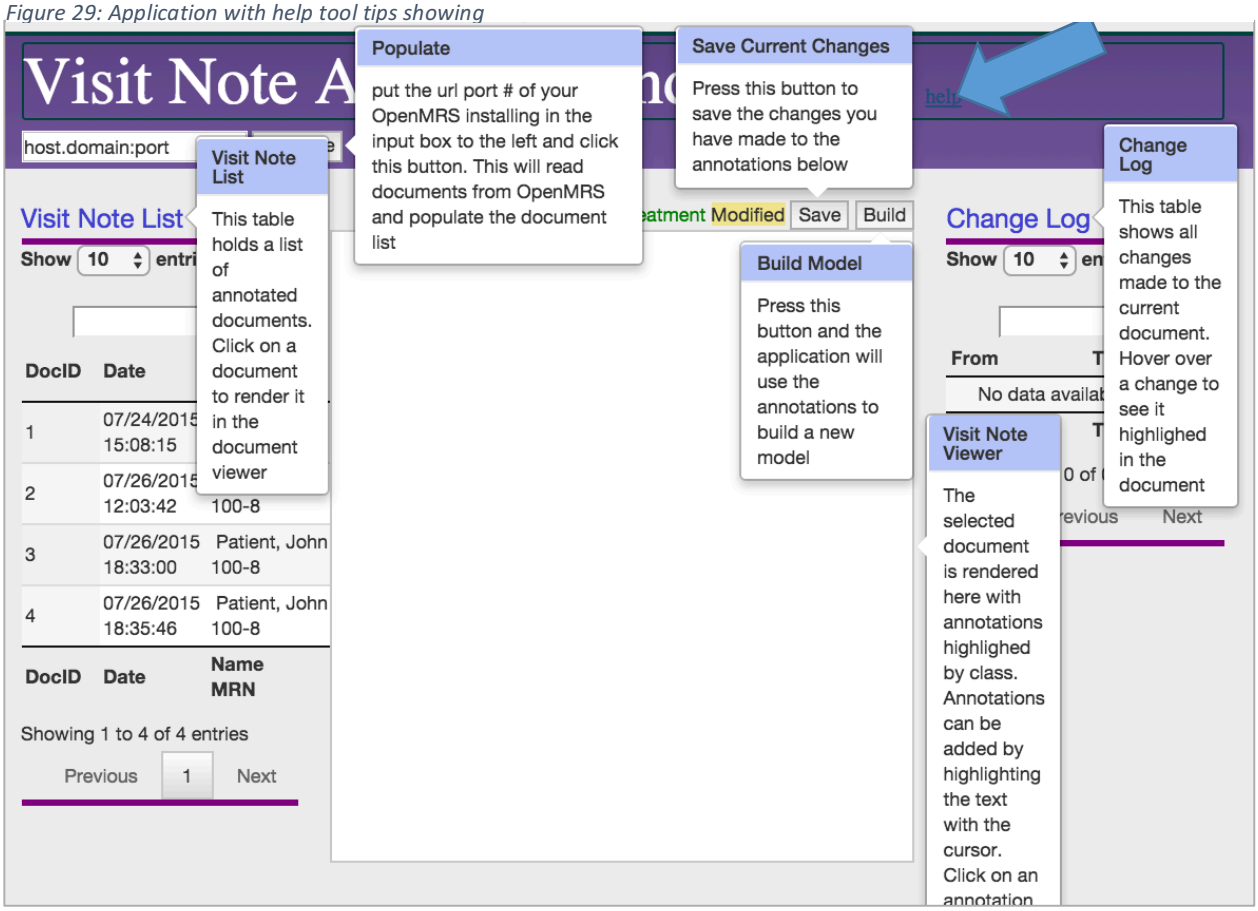
Figure 28: Application with UI components labeled



4.1 Help

Clicking on the help button shows six popover tool tips to assist the user. The next time the user clicks the mouse, these popovers disappear. Figure 29 shows this.

Figure 29: Application with help tool tips showing



4.2 Populate

The Populate component is used to communicate with the OpenMRS installation to read in Visit Notes. More details are provided in section 5.

4.3 Visit Notes List

The Visit Notes list shows the visit notes that were imported from OpenMRS. This list is used to select a document to annotate. Three data points are provided for each Visit Note: The DocID, The date/time of the note, Patient name/Medical Record Number. When a note is selected for editing, it will be highlighted in the Visit Note List

4.4 Visit Note Annotation Editor

The editor allows the user to add new annotations to the Visit Note text as well as correct or delete existing annotations.

4.5 Change log

The Change Log keeps a record of changes made to the current Visit Note being displayed with the option to undue the changes.

4.6 Controls

The controls allow the user to save current changes as well as build the new model based on all documents an all annotations currently in the app.

Visit Note Annotator Usage

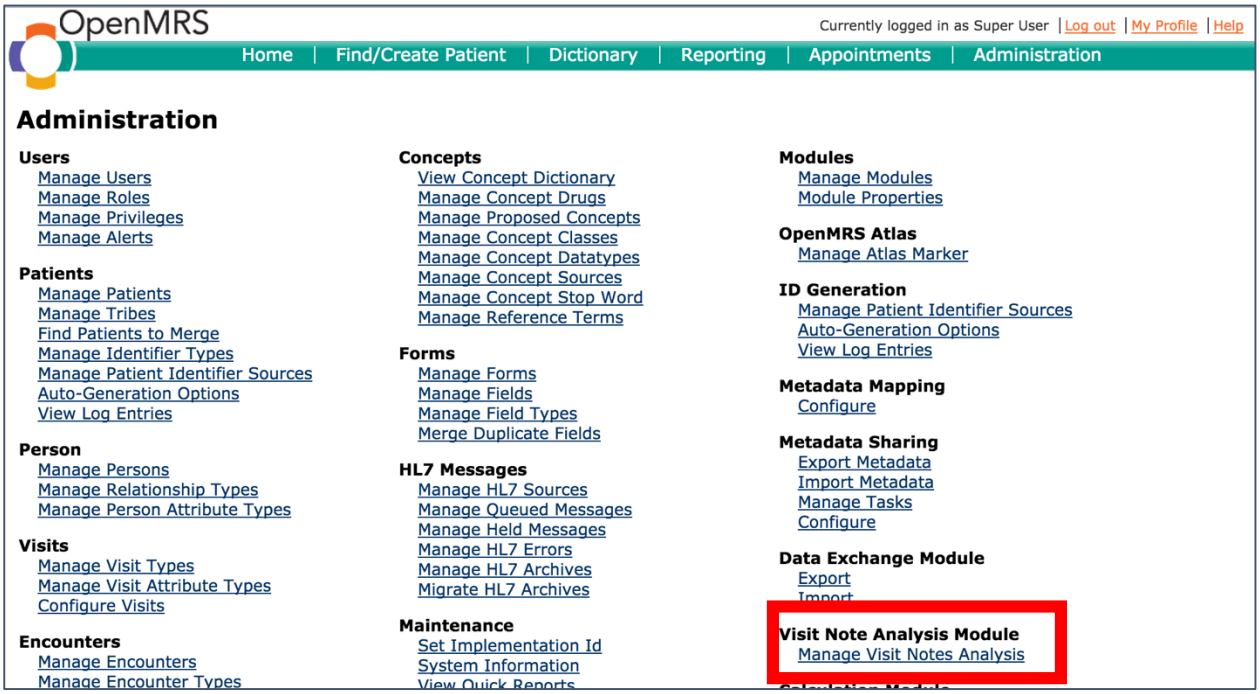
5.1 Import Data from OpenMRS

This application does not read from the OpenMRS database directly. Once the Visit Notes Analaysis module is installed, an HTTP URL is exposed when an administrator navigates to the Manage Visit Notes Analysis page. The web app then uses this endpoint URL to request the Visit Note Data from OpenMRS.

Steps to populate the Visit Notes List are the following:

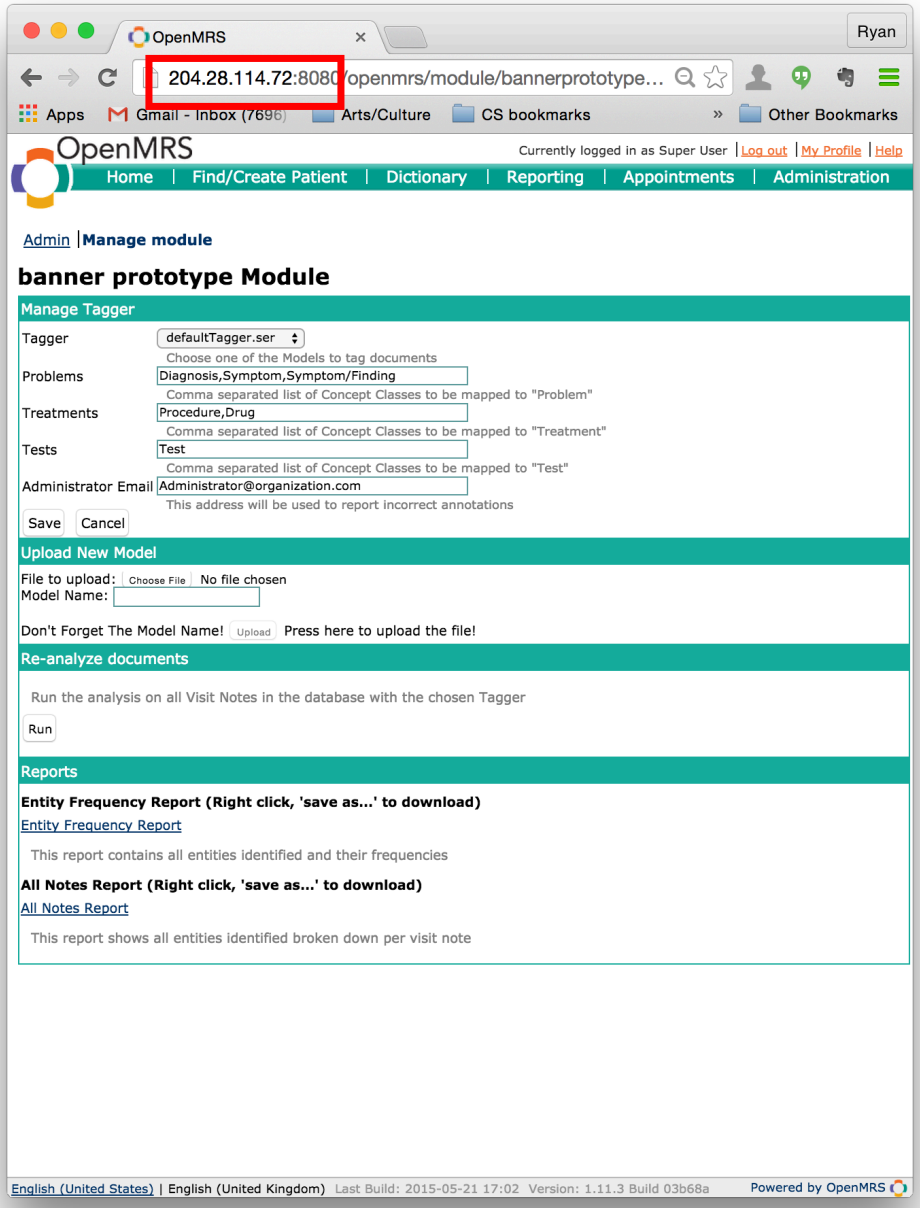
1. Administrator navigates to Manage Visit Notes Analysis page within OpenMRS

Figure 30: OpenMRS Administration page with Manage Visit Note Analysis highlighted



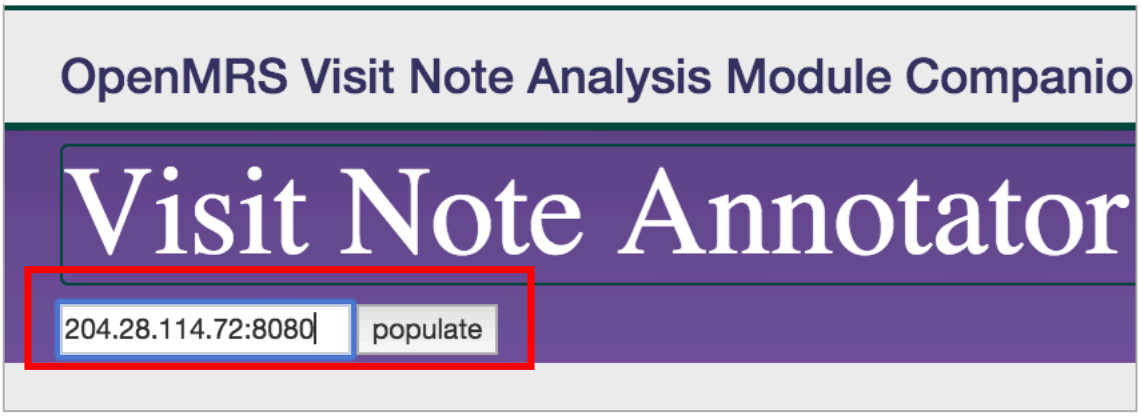
2. Once here, the URL endpoint can be accessed by the Visit Note Annotator and Trainer.

Figure 31: Manage Visit Notes Analysis page, with URL:port# of installation highlighted



3. Enter the URL and port number into the input box next to the “populate” button in the web applications. The URL and Port number are highlighted in figure 31.

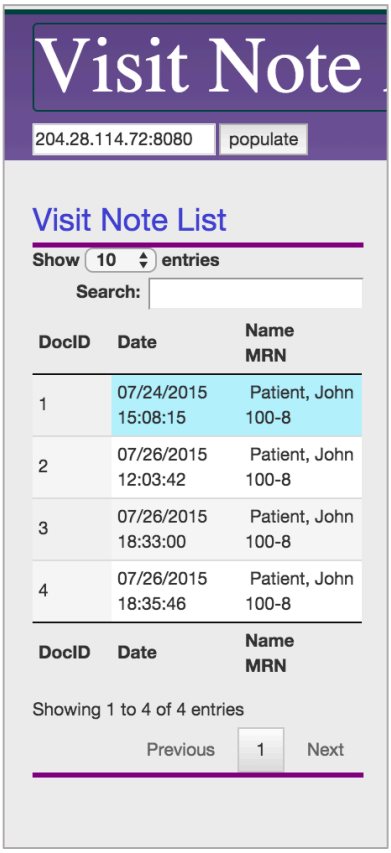
Figure 32: Visit Note Annotator and Trainer app with OpenMRS URL:Port# in "populate" input



4. Click the “populate” button.

5. The documents from OpenMRS will now appear in the Visit Note List, figure 33.

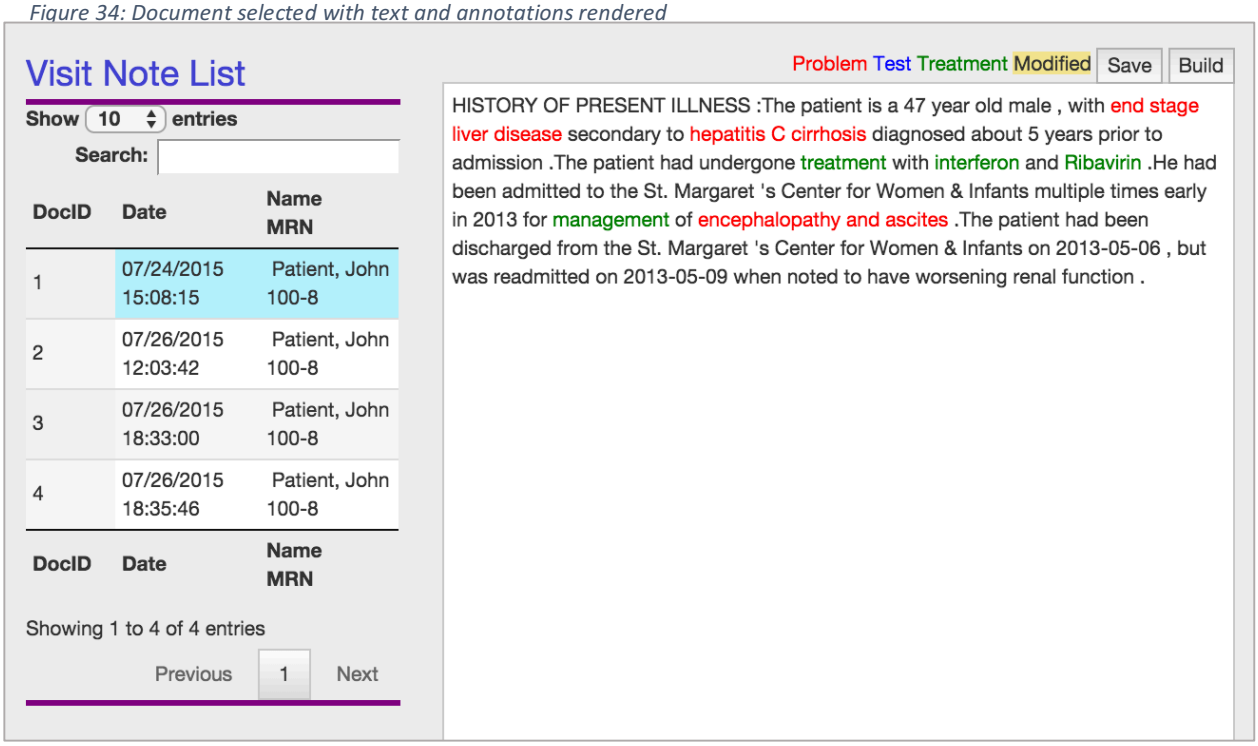
Figure 32: visit notes populated from OpenMRS



5.2 Selecting a Visit Note

The user can search for a specific visit note with the Search Box or file through them sequentially with the “previous” and “next” buttons. Once the desired document is in view, clicking on the document will render it in the Editor.

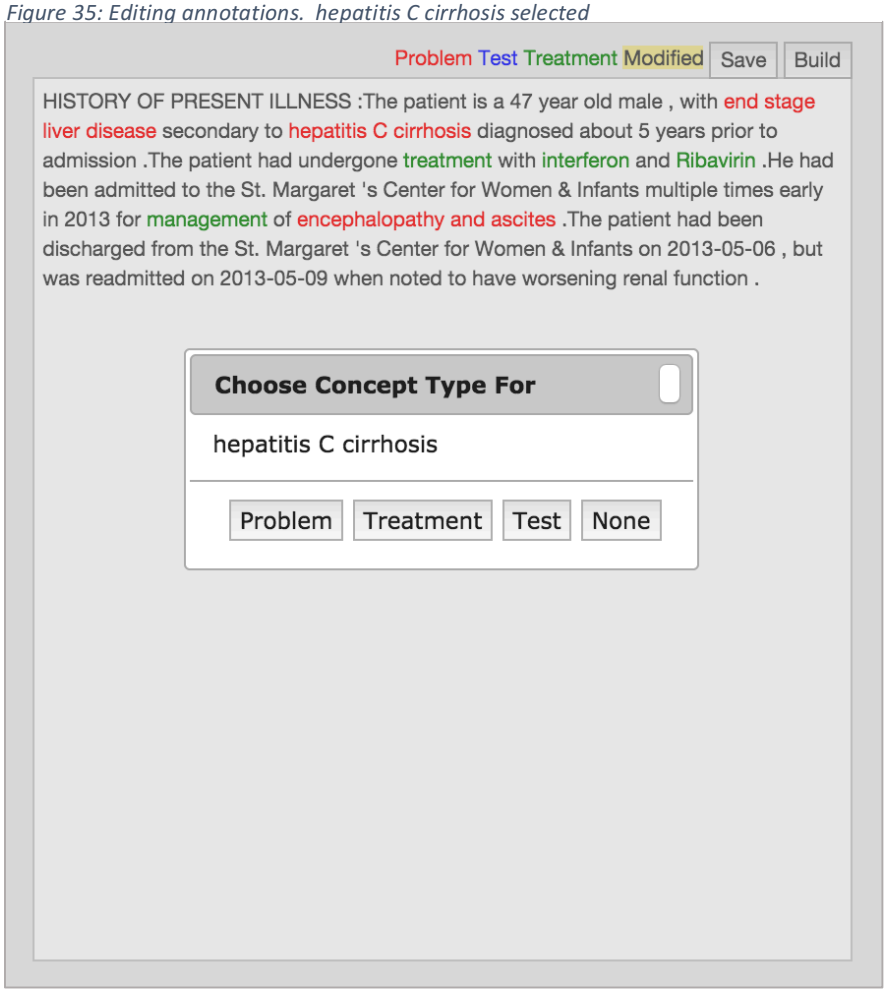
Figure 34: Document selected with text and annotations rendered



5.3 Editing Visit Note Annotations

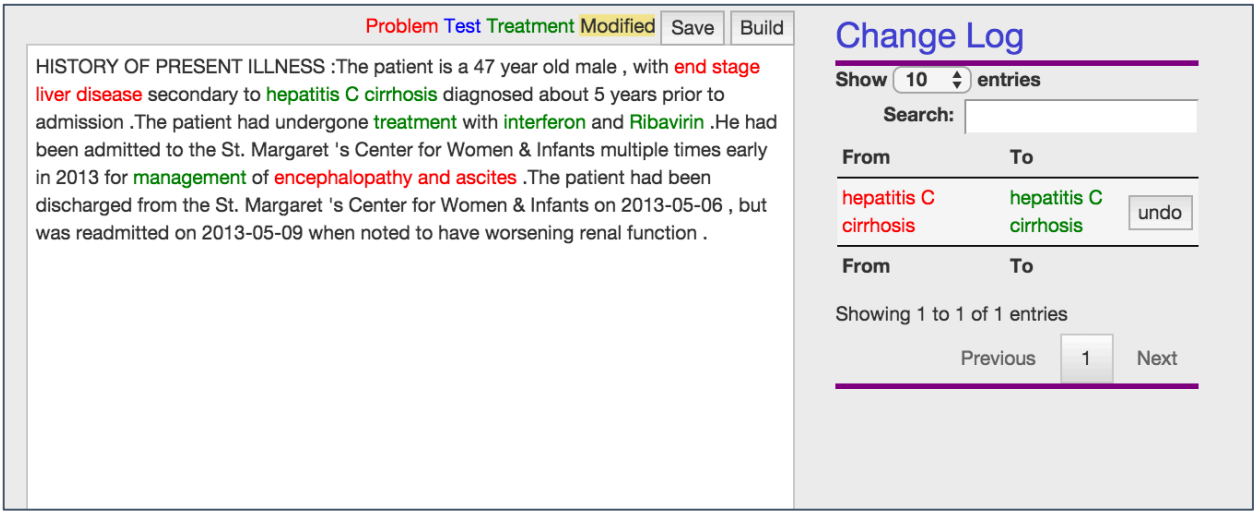
To edit the annotations of a Visit Note, the user can click on an annotation to either change the annotation type, or remove it from the annotations list, figure 35.

Figure 35: Editing annotations. hepatitis C cirrhosis selected



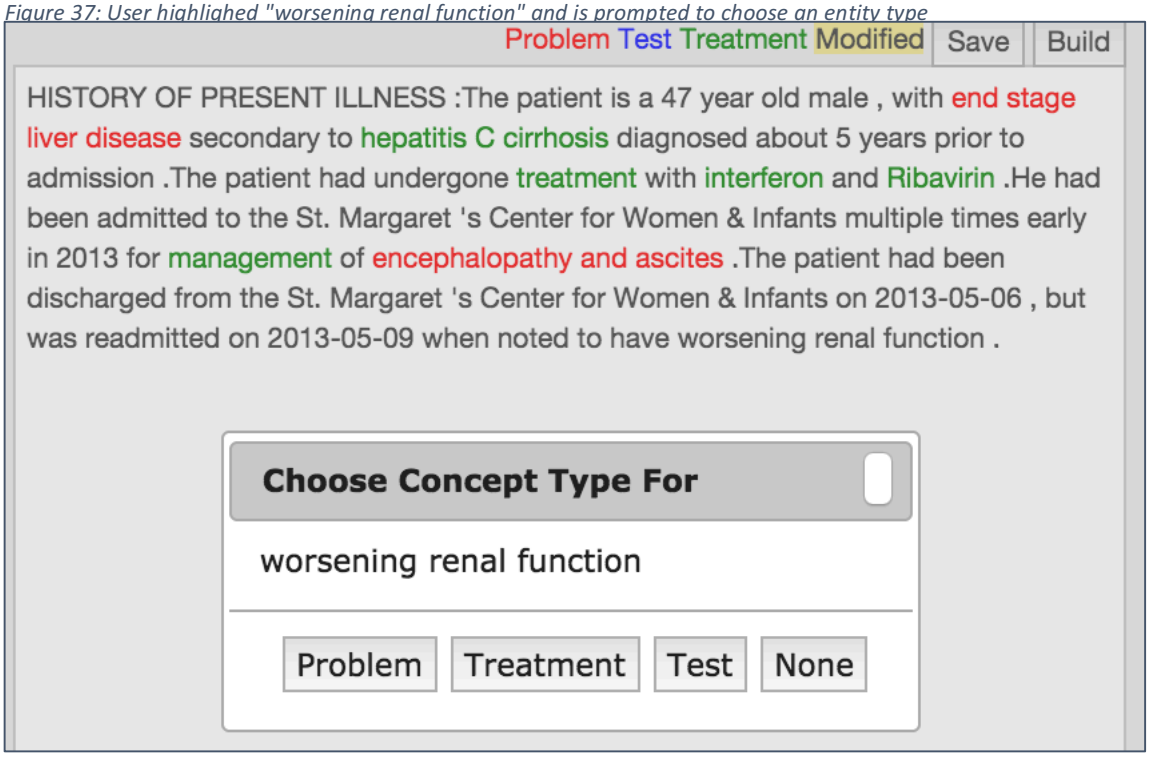
Any changes made by the user will be reflected in the Visit Note rendering as well as the Change Log. Here the user changed the concept type of hepatitis C cirrhosis from “problem” to “treatment.” See figure 36.

Figure 36: user changed the type of "hepatitis C cirrhosis" from problem to test



The user can add a new annotation by highlighting the text. Once a text is highlighted, the same entity-type selection box will appear, allowing the user to select the annotation type, figure 37.

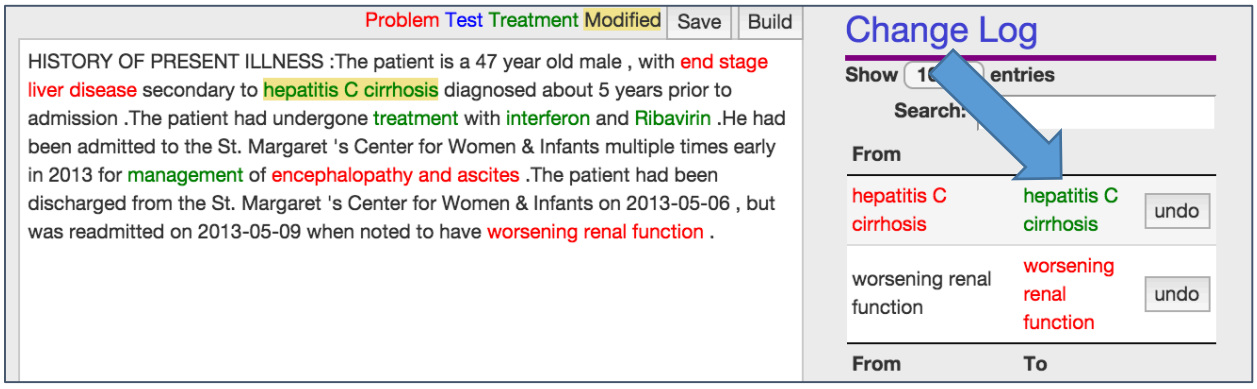
Figure 37: User highlighed "worsening renal function" and is prompted to choose an entity type



5.4 Reviewing Edits

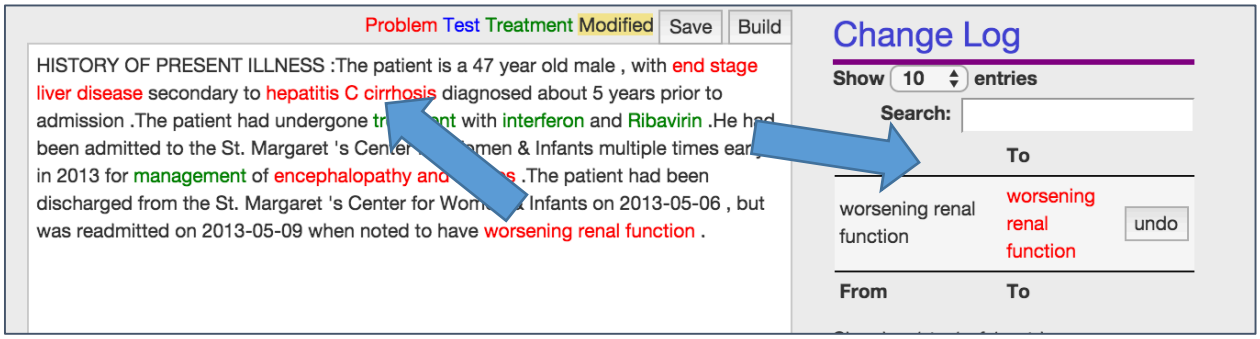
The user can use the change log to review and undo annotation edits that have been made. Hovering the mouse over a change highlights it in the text, figure 38.

Figure 38: User hovers mouse over Change Log entry, corresponding entity is highlighted



Selecting the “undo” button will reverse the corresponding edit change.
Here the user has selected “undo” and “hepatitis C cirrhosis” has reverted from the “treatment” to ”problem” and the entry has been removed from the Change Log. Figure 39

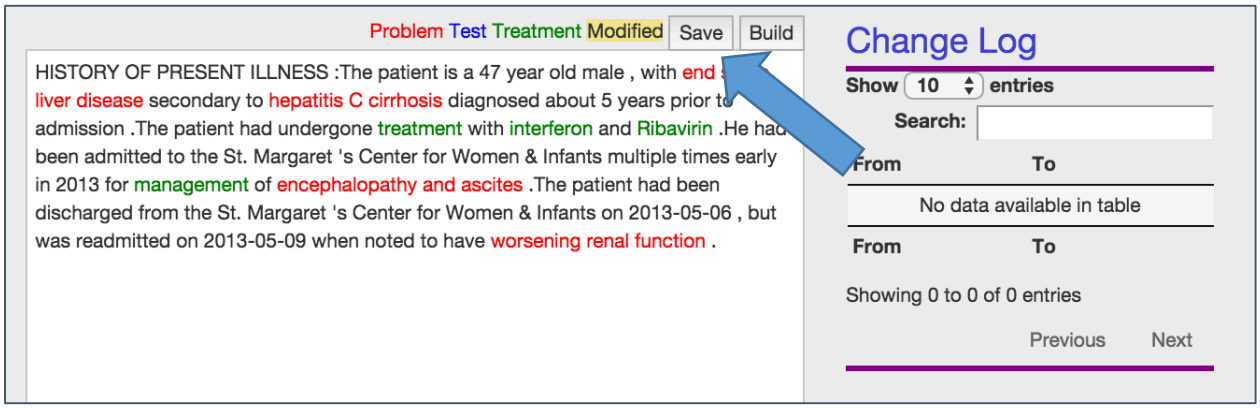
Figure 39: User selects "undo" and hepatitis C cirrhosis is revered from a treatment back to at test. The change log entry is



5.5 Save Annotations

When the user is satisfied with the changes made to the text, they can save the changes by clicking on the save button. If the user navigates to another note without saving, they will be prompted to save.

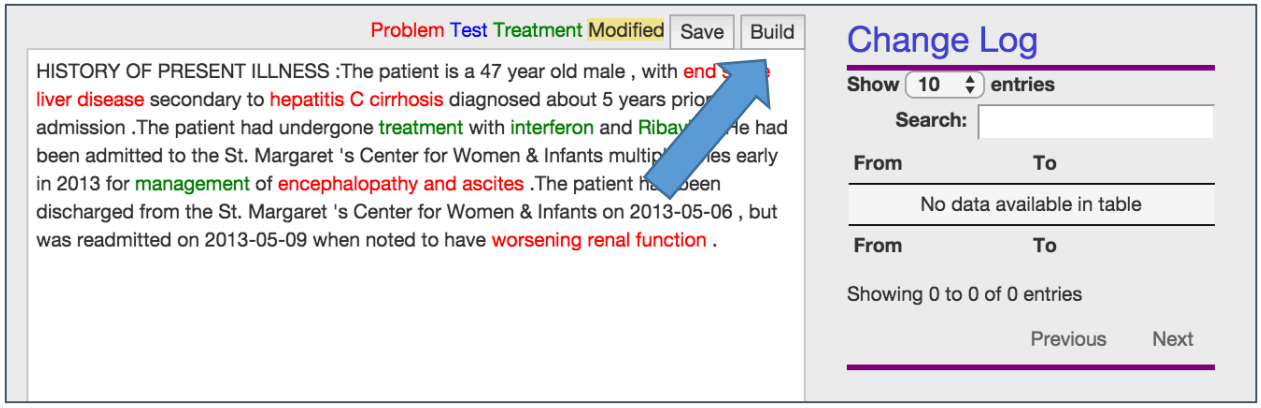
Figure 40: User has saved annotations.



5.6 Build Model

When the user is satisfied with all the annotations on all the documents under scrutiny. They can use the documents and annotations to build a new model. Clicking on the build button will begin model computation, figure 41.

Figure 41: Select "build" to use text and annotations to compute new model.

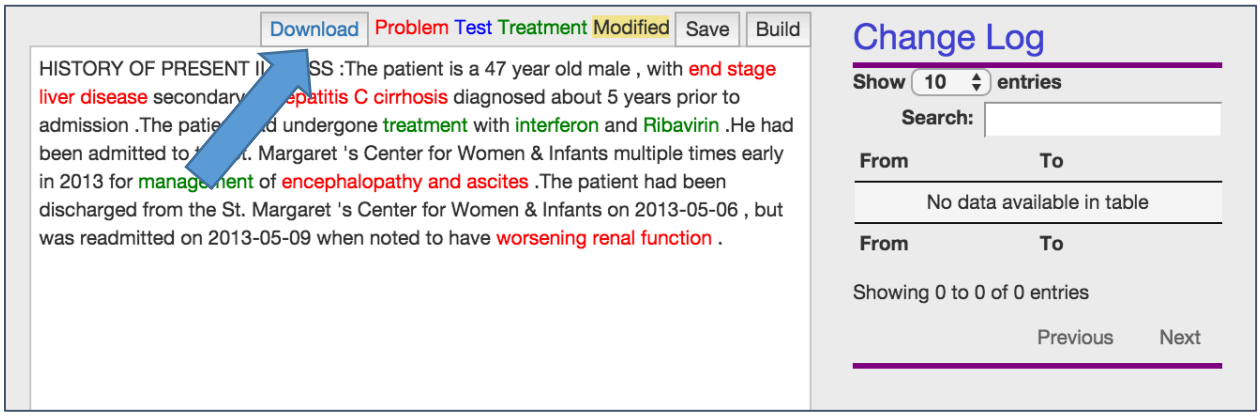


NOTE: This may take some time, depending on the number of documents

5.7 Export Model

When the training process is complete, the user will be notified and a "download" button will appear. Clicking the download button will allow the user to download the new model.

Figure 42: The model has been computed and is available for download.



The user can now refer to section 3 of this guide for instructions on how to load this model into OpenMRS and use it for tagging Visit Note text.