Anchor Finding Interface V1.0

Yoni Halpern

November 13, 2014

Hi! This document accompanies the initial release of the anchor interface described in: "Using Anchors to Estimate Clinical State without Labeled Data" by Y. Halpern, Y.D. Choi, S. Horng, D. Sontag. To appear in the American Medical Informatics Association (AMIA) Annual Symposium, Nov. 2014. It is probably woefully incomplete, please direct questions/comments to Yoni Halpern: halpern@cs.nyu.edu.

I'm happy to work with you to get this up and running on your dataset of interest!

1 Quick Start

1.1 Installation

Pull the github repository for AnchorExplorer with the following command:

```
$ git clone ..
$ cd anchorExplorer
```

We have tested the interface on a system with the following properties:

- Mac or unix OS
- python 2.6 or 2.7
- numpy 1.8.1 and scipy 0.13.3
- scikit-learn 0.15
- networkx 1.8.1
- Tkinter Revision 81008
- ttk 0.3.1

Required python packages are listed a file, requirements.txt and can be installed using pip:

```
$ pip install -r requirements.txt
```

1.2 Data

To start, you need to have patient records in xml format. However, if you want to test that you can get things up and running with dummy data you can use the following command to generate 1000 properly formatted "random" patient records and store them in patients.xml.

```
$ python generate_patients.py 1000 > patients.xml
```

A more realistic dataset can be created if you have access to the MIMIC-II clinical notes.

```
$ python generate_from_mimic.py 1000 > patients.xml
```

1.3 Settings

An example settings file is provided in the examples/ directory. To customize this file for your own data, see section 1.8.

1.4 Preprocess Data

Use the preprocess_patients.py script to preprocess the data in patients.xml for easy lookups and storage. The interface will read from the files generated by this script. Using the example settings file:

```
$ python preprocess_patients.py 1000 patients.xml examples/settings.xml
```

This script does some simple negation detection, bigram detection and stopword removal. If you wish to use your own custom language processing pipeline, see section **??**.

1.5 Run the interface

To run the interface using the example settings file:

```
$ python gui.py examples/settings.xml
```

1.6 Exploring the data

1.7 Learning a model

1.8 Customizing the settings.xml file

The settings.xml file makes it easy[1] to customize the interface for your particular dataset. An example settings.xml file is provided along with the interface code. Most of it can be left as is. The most important parts to customize are the `dataTypes` and `displaySettings` sections.

[1] maybe

1.8.1 patientSets

This section describes which patients should be selected for visualization. If you have many patients, the interface may run slowly, here you can show that you only want to use the first 10,000 patients for an initial exploration.

```
1. <patientSets>
2.    <set name='train' start='0' end='10000'/>
3. </patientSets>
```

The only important patientSet in this version of the code is the "train" set. The start and end fields indicate indices of patients as they are listed in the file visitIDs which is generated by the preprocessing script.

1.8.2 dataTypes

We divide data into different *types* (e.g. text, age, sex, medications, diagnoses, procedures, etc.). Each type is denoted by a `datum` tag and contains a number of `field` tags that describe where this data appears in a patient xml representation.

For example, here is a sample patient representation in xml format:

```
//patient.xml
1. <visit>
2.    <index>qVwLYjLKqlkZhvkf</index>
3.    <MDcomments> SOME TEXT</MDcomments>
4.    <Age> 44 </Age>
5.    <Sex>M</Sex>
6.    <ChiefComplaint> SOME TEXT</ChiefComplaint>
7.    <Diagnosis>
8.        <D_name>HYPERTENSIVE RETINOPATHY</D_name>
9.        <D_code>362.11</D_code>
10.       <D1_name>CHILD/ADULT ABUSE BY SIBLING</D1_name>
11.       <D1_code>E967.5</D1_code>
12.    </Diagnosis>
13.    <TriageAssessment>SOME TEXT</TriageAssessment>
14. </visit>
```

Important: Every patient instance must be enclosed in a `visit` tag and must have a unique `index` tag. All other fields are customizable.

Here is a section of settings.xml that could be used for patient records with this schema.

```
//settings.xml
1. <datum type='text' heirarchy='' prefix=''>
2.     <field name='MDcomments' path='.'/>
```

```
3.     <field name='ChiefComplaint' path='.'/>
4.     <field name='TriageAssessment' path='.'/>
5. </datum>
```

Line 1 of settings.xml says that there is a type of data called **text**. That it is not hierarchical and that it should not be represented with any special prefix.

Lines 2-4 show where in the patient records it can be found (ie. the MDcomments, ChiefComplaint and TriageAssessment sections). The final representation of the **text** data will be a concatenation of these three fields.

1.8.3 Display Settings

For each type of data described in Section 1.8.2, you may want to customize where it is displayed (or whether it is displayed at all). Here you can specify what appears in the `patientSummary` section and the `detailedDisplay`.

The following snippet says that Age, Sex and ChiefComplaint should be displayed as the patient summary:

```
<patientSummary>
    <displayFields>
        <field name='Age'/>
        <field name='Sex'/>
        <field name='ChiefComplaint'/>
    </displayFields>
</patientSummary>
```

And the following snippet says that ChiefComplaint, TriageAssessment, MDcomments and Diagnosis should be displayed in the detailed patient description:

```
<detailedDisplay>
    <displayFields>
        <field name='ChiefComplaint' path='.'/>
        <field name='TriageAssessment' path='.'/>
        <field name='MDcomments' path='.'/>
        <field name='Diagnosis' path="."/>
    </displayFields>
</detailedDisplay>
```

The resulting display is shown in Figure 1.

2 Using the anchor interface

2.1 Starting the interface

The following command starts the interface running:

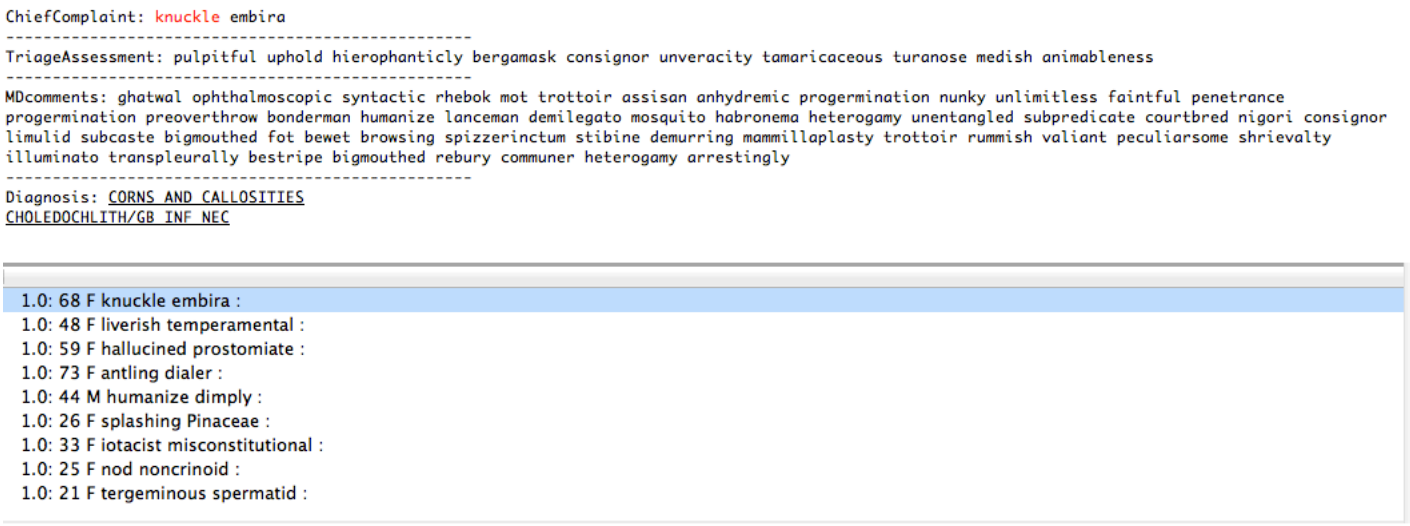

Figure 1: Customizing the patient display as described in section 1.8.3. The bottom list view shows Age, Sex and Chief Complaint for every patient. The top detailed view of a single patient shows fields: ChiefComplaint, TriageAssessment,MDcomments and Diagnosis. We display random words instead of real patient notes.

```
$ python gui.py
```

2.2 Concept Selection Menu (Top Left)

Allows you to select concepts, and create new ones. Some commands:

- New concept button to create a new concept
- '-' delete concept
- 'r' rename concept

2.3 Anchor Display (Top Middle)

Displays the current anchors for the selected topic. Commands:

- Enter text into text box and push enter to add a new anchor
- '-' delete selected anchor
- Anchors can also be added using the anchor suggestion pane (Top Right)

2.4 Anchor Suggestion Pane (Top Right)

Displays suggestions for anchors and allows for navigation and adding of structured anchors. Commands:

- "+" to add selected anchor suggestion
- "suggest" pane gives ranked anchor suggestions.
- other panes give an interface to navigate structured data types (e.g. code, med, etc)

2.5 Learn Button

Learns a model using the currently selected anchors.

2.6 Detailed Patient Display (Middle Right)

Detailed display of the patient selected in Patient List (Bottom Right). Anchors are highlighted red. Structured data types can be clicked to navigate directly to the relevant part of the hierarchy (displayed in Anchor Suggestion Pane).

2.7 Patient List (Bottom Right)

List of patients with summary information. These patients can be filtered using the radio buttons on the left (e.g. view all anchored, view not anchored, etc). After the concept has been "learned" using the Learn Button, this list will be ranked in order of how highly the patient fits the currently selected concept.

Usage:

- arrow keys to navigate
- "+" mark patient as a positive example
- "-" mark patient as a negative example
- "0" remove patient markings

2.8 Structured data representations

We provide a structured representation for ICD9 codes, but if you have other structured representations that you wish to use (NDC, LOINC, etc), you can provide structure files here. More details on this to come.

3 More Information

Contact: Yoni Halpern - halpern@cs.nyu.edu