Key notebooks, files and data demonstrating use of deepdive for RDoC categorization.

Charles Carey (charlieccarey@gmail.com)

March 31, 2016

Contents

Abstract

This document serves as an overview of documentation, methods and assets for DeepDive applications aimed at categorization of literature according to NIMH Research Domain Criteria construct available for RDoC categorization.

The accompanying document **rdoc_report_20160331** will help in understanding the context of the material included in the documentation herein.

1 Important Notebooks

These notebooks contain code and examples demonstrating acquisition of data, preparation of data for DeepDive, creation of and running of DeepDive apps and generation of reports following DeepDive runs.

All notebooks are Jupyter notebooks and were created running the python 2.7.10 kernel.

1.1 Goal- Fetch and manage relevant documents or text from pubmed

- 8_0_1_fetch_random_human_abstracts.ipynb
  - Demonstrates biopython's entrez module to search pubmed by phrases, keywords, dates etc.
  - Demonstrates retrieval of pubmed ids from such searches.
  - Demonstrates fetching of pubmed abstracts and metadata using python's medic package.
  - Describes alternative data sources for abstracts and full text (Appendix)

1.2 Goal- Manage distribution of documents to annotators for annotation or scoring and retrieve versions of abstracts into medic.

- 9_2_round_6_annotations_out.ipynb
  - Demonstrates use of Biopythons Bio.Entrez to interface to NCBI Entrez utilities to obtain pubmed ids matching search criteria.
  - Demonstrates use of Url2PubmedPmcPdf.py for programatically fetching PDFs from pubmed central
  - Demonstrates naming and distribution of articles for annotation.
  - Demonstrates fetching the corresponding pubmed abstract and metadata into medic

Note medic was not available in python 3 at this time.

1.3 Goal- Extract highlighted words from PDFs.

- mine_pdf_annotations.ipynb
  - Demonstrates development and use of python pdfminer package to manipulate and extract features of PDFs including highlighting.
- 5_0_1_semi_automated_match_annotations_into_plain_text_abstracts.ipynb and 2_spacy_parse_annotations.ipynb
  - Demonstrates use of Spacy to recognize and split paragraphs (abstracts) into single sentences.
  - Demonstrates opening versions of abstract with annotation obtained from PDF side by side with version of abstract obtained as raw text from Medic.
  - The abstract text from the PDFs often include acronyms.
  - The abstract text obtained from Medic often has the acronyms fully expanded to their full terms.
  - The medic version is therefore preferred due to this expansion of acronyms.
  - Opening these side by side allows us to edit the medic version towards the annotated version, while manually intervening in cases where the PDF mining process was error prone (rearranged sentence ordering, missed words etc.)

- See also extract.org (a plain text org-mode file) exploring python alternatives to PDFMiner.and specific help topics found for using PDFMiner.

1.4 Goal- Use NLP to process text from pubmed:

- 10_3_design_deepdive_wrapper.ipynb
  - Demonstrates generation of NLP processed abstract or text data for simple deepdive apps.
  - Assumes Simple app requires input that is True, False (and Null) labeled.
  - Includes some sketches and unfinished code for better organzing app creation.
  - Discusses medic and medic database tables in more detail.

1.5 Goal- Generate labeled input sentences or text for DeepDive:

- 11_1_aud_per_vs_others_create_input.ipynb
  - Demonstrates consolidating input data into single source files for deepdive apps depending on the class to be predicted.
    - Includes examples compiling 'true' 'false' labeling.
    - Includes examples adding 'pseudonull' labeling of 'false and true' labels in order that all 'trues' and 'falses' are exposed for reporting purposes (otherwise their predictions may be inaccessible in this version of deepdive).
    - Corrects compiled sentences to avoid a common crash condition due to input data issues when running DeepDive.

1.6 Goal- Create, run and report on multiple deepdive apps:

- 11_2_aud_per_vs_others_run_and_summarize_deepdive.ipynb
  - Demonstrates creation of simple apps based on template and template directory.
  - Demonstrates population of app input data from above steps.
  - Demonstrates a variant of cross-validation.
  - Discussses a couple crash conditions

  - Demonstrates SQL required to populate cc_all_predictions.tsv, itself required for generating reports using R.
  - Demonstrates running reports on cc_all_predictions.tsv using report_dd_confusion_matrix.R to obtain confusion matrix and associated stats.
  - Demonstrates generation of plots on cc_all_predictions.tsv using report_dd_confusion_matrix.R for visual inspection of model performance.

1.7 Goal- Describe how to manage input data for DeepDive to work around deficient DeepDive reporting issues.

The DeepDive version v0.7, while providing access to results of the heldout 'test' portion of training data, does not provide an easy way to examine the model with regards to how it performs on the 'training' portion of the training data.

- 10_1_design_deepdive_sql_reports_for_confusion_matrix.ipynb and 10_2_design_deepdive_confusion_matrix_report.ipynb
  - Demonstrates use of 'pseudo-null' labeling of the training set in order to expose a prediction on the dataset.
  - Discusses some of the issues related to SQL extraction of the 'training' 'heldout test' and 'other unlabeled-to-be-predicted' from our hack to get the reporting we wanted.

1.8 Goal - Do cross validation with deepdive

This is not true k-fold Cross validation, because we are resampling with replacement in each round.

- 12_example_cross_validations.ipynb
  - Demonstrates cross validation.
  - Briefly discusses why a standard k-fold cross validation would be more difficult with DeepDive.
  - Plots a few stats from the cross-validation.

2 Important Scripts

2.1 scripts/bag_of_words_parsing.py

- (Also included in Appendix in 10_3_design_deepdive_wrapper.ipynb.)
- Uses Python spaCy library for Natural Language Processing tasks.
- Use to get lemmatized bag of words, or biwords from text.
- Use to generate raw and annotated ('t' 'f' or 'Null' labeled) sentences.

2.2 scripts/plot_deepdive_calibration.R

- Plots a slightly easer to understand version of DeepDive's calibration plots.
- Uses table that DeepDive itself created for its own calibration plotting.

2.3 scripts/report_dd_confusion_matrix.R

- Reads a custom TSV formatted file we obtained from SQL of DeepDive results.
- Uses R caret to report confusion matrix and compute confusion matrix statistics such as Precision, Accuracy, Specificity.
- Uses R ggplot2 library to generate scatter plot like pdfs of training, test and unknown data to graphically view deepdive app model performance.

2.4 scripts/Url2PubmedPmcPdf.py

- Uses Biopython's Bio.Entrez module to try to convert pubmed_ids to URLs that can be downloaded programatically from pubmed central.

3 Other Scripts and programming notes.

- pubmed_search_to_rdoc_db.py
  - An abandomed sketch in which we were going to manage all our pdfs in a database rather than in system folders.
- extract.org Some scraps related to finding solutions to extract highlighted sections from PDFs.

4 Important Templates

- Template for prediction of membership in class based on a simple bag of words based deepdive app.

```
#ls -c1 templates_deepdive_app_bagofwords/*
templates_deepdive_app_bagofwords/db.url  # Sets name of deepdive app.
templates_deepdive_app_bagofwords/deepdive.conf  # Configures tables and
                                                 # extractors for deepdive app.
templates_deepdive_app_bagofwords/schema.sql  # Table schema for input data for deepdi

templates_deepdive_app_bagofwords/input:
init.sh  # Processes deepdive app input data when deepdive initdb is run.

templates_deepdive_app_bagofwords/udf:
dd_extract_features.py  #  User defined function (UDF) referred to within deepdive.con
```

5 Important Datasources

5.1 TODO PDFs (Unannotated or Annotated)

5.2 TODO Processed records from Annotated PDFs.

5.3 Data sources specifically for DeepDive input folder.

These data sources are used to construct input data for simple 'is category' type of DeepDive apps based on bags of lemmatized words.

- /tasks/task_data_sentences
  - Category labeled raw or NLP parsed text data ready for deepdive app input.
  - all_sentences/*
    - labeled data that may be selected and combined based on the particular app
  - ./xx_vs_yy/raw_sentences
    - Fields are:
      - PMID, abstract, category label (t,f, N), null field (just in case we'll use this in deepdive).
    - sentences ready for populating _raw_sentences table in our simple deepdive apps.

    - Not really used for much other than access to original sentences within deepdive app from other sentences.
- ./xx_vs_yy/annotated_sentences
    - Fields are:
        - PMID, abstract, category label (t,f, N), null field (reserved for deepdive).
    - NLP processed text (lemmas, or biwords etc.)
    - Features are in postgres accessible array format.
    - Ready for populating _annotated_sentences table in our simple deepdive apps.
    - Further processed by our deepdive app into individual features for deepdive to weight and use for class membership training, testing and prediction.

6 Example apps

The following apps are based on DeepDive training using lemmatized bags of words of full abstracts.

Apps should be run within their directory. cd into the app for deepdive initdb and deepdive run because deepdive wants the db.url in the app directory.

6.1 Caution on app names.

Note, the GIBBS Sampler breaks on either very long app names or paths, so don't make names longer than 55 characters or if it was the full path, 150 characters?

This may be the same error or a separate error with the gibbs sampler in which I crashed it running at depth > 1 below my main deepdive_app folder. (I had apps organized by app_type under deepdive_app).

6.2 An example of my app naming convention

The app named **Auditory_146_vs_Arousal_1_1000_pdx_un_Aro_154**

Takes as training data:

- 146 Auditory perception abstracts (processed into bags of lemmas)
- 1000 Arousal abstracts (a 1st set of Arousal abstracts)

- Goal is build a model predicting abstract belongs to Auditory Perception.
- The 1000 Arousal abstracts are a 'non-target' set chosen to help train the model to recognize what is NOT Auditory perception.

Takes as additional data to be predicted:

- A set of 154 unlabeled Arousal abstracts are also included so that we can observe for this non-training data whether the arousal abstracts are predicted into the Auditory Perception class or not (we are hoping to the extent that they are truly different topics that they are not predicted into Auditory Perception).

6.3 Categorize Auditory perception articles based on a small training set for auditory perception and a larger training set of various topics as the non-target.

This group of 4 apps shows relatively poor performance when the target class is small.

- Auditory_146_vs_Arousal_1_1000_pdx_un_Aro_154
- Auditory_146_vs_Arousal_1_1000_pdx_un_Aud_1_1000
- Auditory_146_vs_Disease_1000_pdx_un_Aud_1_1000
- Auditory_146_vs_Psyc_1000_pdx_un_Aud_1_1000

Only 146 abstracts in the target class led to fair performance when trained against a more distant topic 'disease' as non-target. But inferior performance when more similar topics were used as the non-target class.

This is sometimes despite the model looking good on the training and heldout test data. When applied to 'less preselected' Auditory Perception abstracts, the model performed worse. The models were likely overfitted to the particular features of the small training (and therefore also small test) sets.

6.4 Using a larger training set vastly improves the models.

Using 1000 'random' Auditory Perception abstracts to represent the target class, and 1000 non-target abstracts as training data resulted in much improved and generalized performance of the model.

- Auditory_2_1000_vs_Arousal_1_1000_pdx_un_Aro_154
- Auditory_2_1000_vs_Arousal_1_1000_pdx_un_Aud_1_1000
- Auditory_2_1000_vs_Disease_1000_pdx_un_Aud_1_1000
- Auditory_2_1000_vs_Psyc_1000_pdx_un_Aud_1_1000

7 Software versions used.

- DeepDive (v0.7.0-0-gf4e6dfe, build date: 2015-07-14T15:43:32-07:00)
- PostgreSQL (9.4.4)
- R (3.2.3)
- Python (2.7.10)
- medic (2.4.1)
- pdfminer (20140328)
- R library caret (6.0-64)
- R library ggplot2 (2.0.0)