icdcodex Documentation

Release $_{version}{}_{=0.4.5'}$

Jeremy Fisher

Oct 18, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Experimental

This is experimental software and a stable API is not expected until version 1.0

1.1 What is it?

A python library for building vector representations of ICD-9 and ICD-10 codes. Because it takes advantage of the hierarchical nature of ICD codes, it also provides these hierarchies in a `networkx` format.

1.2 Motivation

`icdcodex` was the first prize winner in the Data Driven Healthcare Track of John Hopkins' MedHacks 2020. It was hacked together to address the problem of ICD miscodes, which is a major issue for health insurance in the United States. Indeed, while ICD coding is tedious and labour intensive, it is not obvious how to automate because the output space is enourmous. For example, ICD-10 CM (clinical modification) has over 70,000 codes and growing.

There are many strategies for target encoding that address these issues. `icdcodex` has two features that make ICD classification more amenable to modeling:

- Access to a `networkx` tree representation of the ICD-9 and ICD-10 hierarchies
- Vector embeddings of ICD codes using the node2vec algorithm (including pre-computed embeddings and an interface to create new embeddings)

1.3 Example Code

```
from icdcodex import icd2vec, hierarchy
embedder = icd2vec.Icd2Vec(num_embedding_dimensions=64)
embedder.fit(*hierarchy.icd9())
X = get_patient_covariates()
y = embedder.to_vec(["0010"])  # Cholera due to vibrio cholerae
```

In this case, `y` is a 64-dimensional vector close to other `Infectious And Parasitic Diseases` codes.

1.4 Related Work

- node2vec Paper, Website, Code, Alternate Code
- Learning Low-Dimensional Representations of Medical Concepts: Paper, Code

1.5 The Hackathon Team

- Jeremy Fisher (Maintainer)
- Alhusain Abdalla
- Natasha Nehra
- Tejas Patel
- Hamrish Saravanakumar

1.6 Documentation

See the full documentation: https://icd-codex.readthedocs.io/en/latest/

1.7 Contributions

Contributions are always welcome!

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install icdcodex, run this command in your terminal:

```
$ pip install icdcodex
```

This is the preferred method to install icdcodex, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for icdcodex can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/icd-codex/icd-codex
```

Or download the tarball:

```
$ curl -OJL https://github.com/icd-codex/icd-codex/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

3.1 `networkx` Hierarchy

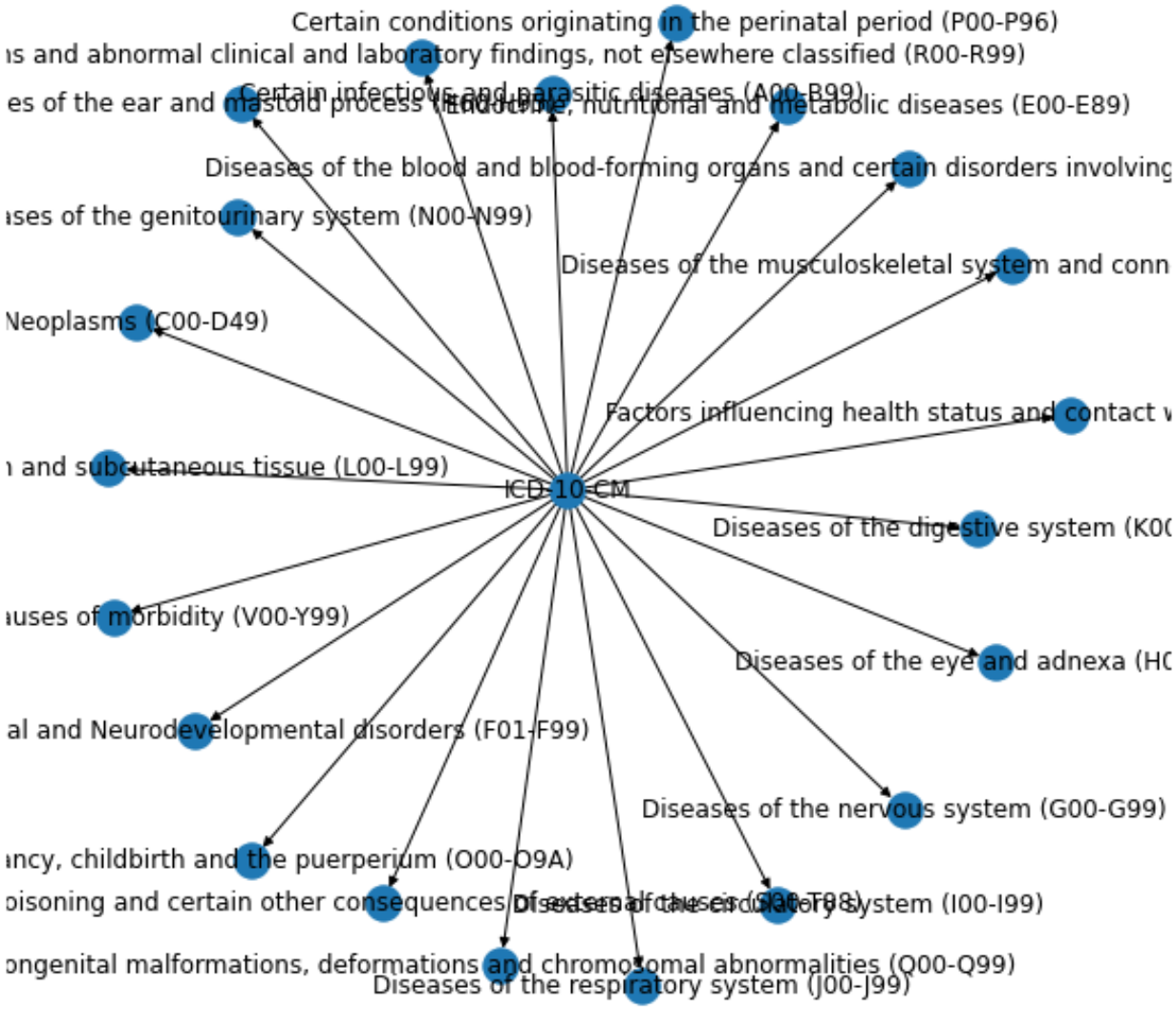
The International Classification of Disease seperates diseases into chapters, subchapters and so on. `icdcodex` considers each of these as a node in a network – and provides access to this representation through `networkx`.

```
from icdcodex import hierarchy
icd_10_cm_hierarchy, icd_10_cm_codes = hierarchy.icd10cm("2020")
```

The network is unstructured, by default. But we can use `networkx` to do graphical analysis. For instance, we can generate a spring layout of the chapters using breadth first traversal.

```
import networkx as nx
from networkx.algorithms.traversal.breadth_first_search import bfs_tree
import matplotlib.pyplot as plt

G = nx.relabel_nodes(icd_10_cm_hierarchy, {"root": "ICD-10-CM"})
G_chapters = bfs_tree(G, "ICD-10-CM", depth_limit=1)
plt.figure(figsize=(8,8))
nx.draw(G_chapters, with_labels=True)
```



3.2 Vector Embeddings

`icdcodex` uses this graphical structure to build a dense representation of individual codes. So far, we provide the `node2vec` algorithm, accessible through the `icd2vec` module.

```
from icdcodex import icd2vec, hierarchy
# workers=-1 parallelizes the node2vec algorithm across all available CPUs
embedder = icd2vec.Icd2Vec(num_embedding_dimensions=2, workers=-1)
embedder.fit(*hierarchy.icd9())
```

Dense representations have the property that similar vectors have similar locations in vector space. For example, `033.0 (Whooping cough due to bordetella pertussis)` and `034.0 (Streptococcal sore throat)` are both bacterial diseases. `910.1 (Abrasion or friction burn of face, neck, and`

`scalp except eye, infected)` is an injury. We expect `034.0` and `033.0` to be closer to one another than to `910.1`

```
codes_of_interest = ["0330", "0340", "9101"]
codes_of_interest_continuous = embedder.to_vec(codes_of_interest)
codes_of_interest_continuous
```

```
array([[ 0.78746617, -1.1355207 ],
       [ 0.7202955 , -0.97387433],
       [ 5.667383  , -0.13091612]], dtype=float32)
```

3.3 Example: Predicting MIMIC-III diagnostic codes

For a more involved example, we'll build a scikit-learn pipeline. To get our data, we'll use MIMIC-III. A demo version can be accessed through the GCP Big Query service by running `ADD DATA > Pin a project > Enter a project name > physionet-data`. Run the SQL query:

```
SELECT
    i.seq_num, i.subject_id, i.icd9_code, j.los, k.gender, k.dob, k.dod, l.admittime
FROM `physionet-data.mimiciii_demo.diagnoses_icd` as i
    INNER JOIN
        `physionet-data.mimiciii_demo.icustays` as j
        ON i.hadm_id = j.hadm_id
    INNER JOIN
        `physionet-data.mimiciii_demo.patients` as k
        ON i.subject_id = k.subject_id
    INNER JOIN
        `physionet-data.mimiciii_demo.admissions` as l
        ON i.hadm_id = l.hadm_id
```

Then, save the results as `data.csv`.

First, we'll import boilerplate data science libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

Then, load the data and do feature engineering to give the model something with which to predict ICD codes.

```
df = pd.read_csv("data.csv").rename(columns={
    "los": "length_of_stay",
    "dob": "date_of_birth",
    "dod": "date_of_death",
    "admittime": "date_of_admission"
})
df["date_of_birth"] = pd.to_datetime(df["date_of_birth"]).dt.date
df["date_of_death"] = pd.to_datetime(df["date_of_death"]).dt.date
df["date_of_admission"] = pd.to_datetime(df["date_of_admission"]).dt.date
df["age"] = df.apply(lambda e: (e['date_of_admission'] - e['date_of_birth']).days/365,
↪ axis=1)
df = df[df.seq_num == 1]  # we limit ourselves to the primary diagnosis code for␣
↪simplicity
df.gender = LabelEncoder().fit_transform(df.gender)
G, icd_codes = hierarchy.icd9()
df = df[df.icd9_code.isin(G.nodes())]
```

(continues on next page)

(continued from previous page)

```
features = ["length_of_stay", "gender", "age"]
X = df[features].values
y = df[["icd9_code"]].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
↪state=42)
```

Now, we can use any regression model that can predict a vector, including `RandomForestRegressor`

```
from sklearn.ensemble import RandomForestRegressor
y_train_continuous = embedder.to_vec(y_train.reshape(-1))
clf = RandomForestRegressor()
clf.fit(X_train, y_train_continuous)
```

CHAPTER

FOUR

REPRODUCING SIRRICE/ICD9

icdcodex recapitulates the functionality of sirrice/icd9 which has similar functionality, which is somewhat dated and does not support ICD-10

```
import networkx as nx
from icdcodex import hierarchy
G, codes = hierarchy.icd9()
```

4.1 A simple demonstration

From the read me

> The library encodes ICD9 codes in their natural hierarchy. For example, "Cholera due to vibrio cholerae" has the ICD9 code 001.0, and is categorized as a type of Cholera, which in turn is a type of Intestinal Infectious Disease. Specifically, 001.0 has the following hierarchy: "Cholera due to vibrio cholerae" has the ICD9 code 001.0, and is categorized as a type of Cholera, which in turn is a type of Intestinal Infectious Disease.

We can find this hierarchy by using the shortest_path method

```
cholerae_icd_code = "001.0".replace(".", "")
root_node, *natural_hierarchy = nx.shortest_path(G, source="root", target=cholerae_
↪icd_code)
natural_hierarchy
```

```
['Infectious And Parasitic Diseases',
 'Intestinal Infectious Diseases',
 'Cholera',
 '0010']
```

4.2 Using the library

4.2.1 Find top level codes

To find the top level codes, we can do a one layer traversal starting at the root.

```
from networkx.algorithms.traversal.breadth_first_search import bfs_tree
top_level_nodes = bfs_tree(G, source="root", depth_limit=1)
top_level_nodes.nodes()
```

```
NodeView(('root', 'Infectious And Parasitic Diseases', 'Neoplasms', 'Endocrine,␣
↪Nutritional And Metabolic Diseases, And Immunity Disorders', 'Diseases Of The Blood␣
↪And Blood-Forming Organs', 'Mental Disorders', 'Diseases Of The Nervous System And␣
↪Sense Organs', 'Diseases Of The Circulatory System', 'Diseases Of The Respiratory␣
↪System', 'Diseases Of The Digestive System', 'Diseases Of The Genitourinary System',
↪ 'Complications Of Pregnancy, Childbirth, And The Puerperium', 'Diseases Of The␣
↪Skin And Subcutaneous Tissue', 'Diseases Of The Musculoskeletal System And␣
↪Connective Tissue', 'Congenital Anomalies', 'Certain Conditions Originating In The␣
↪Perinatal Period', 'Symptoms, Signs, And Ill-Defined Conditions', 'Injury And␣
↪Poisoning', 'Supplementary Classification Of External Causes Of Injury And Poisoning
↪', 'Supplementary Classification Of Factors Influencing Health Status And Contact␣
↪With Health Services'))
```

Any arbitrary sub-nodes are obtained in a similar fashion

```
intestinal_infectious_disease_nodes = bfs_tree(G, source="Intestinal Infectious␣
↪Diseases").nodes()
intestinal_infectious_disease_nodes
```

```
NodeView(('Intestinal Infectious Diseases', 'Cholera', 'Typhoid and paratyphoid fevers
↪', 'Other salmonella infections', 'Shigellosis', 'Other food poisoning (bacterial)',
↪ 'Amebiasis', 'Other protozoal intestinal diseases', 'Intestinal infections due to␣
↪other organisms', 'Ill-defined intestinal infections', '0010', '0011', '0019', '0020
↪', '0021', '0022', '0023', '0029', '0030', '0031', '00320', '00321', '00322', '00323
↪', '00324', '00329', '0038', '0039', '0040', '0041', '0042', '0043', '0048', '0049',
↪ '0050', '0051', '0052', '0053', '0054', '00581', '00589', '0059', '0060', '0061',
↪'0062', '0063', '0064', '0065', '0066', '0068', '0069', '0070', '0071', '0072',
↪'0073', '0074', '0075', '0078', '0079', '00800', '00801', '00802', '00803', '00804',
↪ '00809', '0081', '0082', '0083', '00841', '00842', '00843', '00844', '00845',
↪'00846', '00847', '00849', '0085', '00861', '00862', '00863', '00864', '00865',
↪'00866', '00867', '00869', '0088', '0090', '0091', '0092', '0093'))
```

4.2.2 Find all nodes by a search criteria

```
[n for n in G.nodes() if n.startswith("001")]
```

```
['0010', '0011', '0019']
```

4.2.3 Find all codes (i.e., leaf nodes) by a search criteria

```
cholerae_nodes = bfs_tree(G, source="Cholera").nodes()
[n for n in cholerae_nodes if G.degree[n] == 1]
```

```
['0010', '0011', '0019']
```

4.2.4 Get the description of a code

```
G.nodes()["0010"]
```

```
{'description': 'Cholera due to vibrio cholerae'}
```

4.2.5 Get a nodes parent and siblings

```
parent, = G.predecessors("0010")
print(f"parent: {parent}, siblings: {G[parent]}")
```

```
parent: Cholera, siblings: {'0010': {}, '0011': {}, '0019': {}}
```

CHAPTER

FIVE

ICDCODEX

5.1 icdcodex package

5.1.1 Subpackages

icdcodex.data package

Module contents

submodule for holding data artifacts, including ICD9/10 hierarchys in serialized networkX format and precomputed embeddings

5.1.2 Submodules

5.1.3 icdcodex.datacleaning module

preprocess icd-10 hierarchy into a graphical structure that node2vec can use

icdcodex.datacleaning.**build_icd10_hierarchy**(*xml_root: untangle.Element*, *codes: List[str]*, *root_name: Optional[str] = None*, *prune_extra_codes: bool = True*)

build the icd10 hierarchy

Some codes are specified to be invalid by plain text, so they are pruned by comparing them to a specified set of codes.

Parameters

- **xml_root** (*untangle.Element*) – root element of the code table XML
- **codes** (*List[str]*) – list of ICD codes
- **root_name** (*str, option*) – arbitrary name for the root of the hierarchy. Defaults to "root."
- **prune_extra_codes** (*bool*) – If True, remove any leaf node not specified in *codes*

Returns icd10 hierarchy and ICD-10-CM codes

Return type Tuple[nx.Graph, List[str]]

icdcodex.datacleaning.**build_icd10_hierarchy_from_url**(*code_desc_url*, *code_table_url*, *root_name: Optional[str] = None*, *return_intermediates=False*)

build the icd10 hierarchy by downloading from cms.gov

Parameters

- **code_desc_url** (*str*) – url to the "Code Descriptions in Tabular Order (ZIP)" file
- **code_table_url** (*str*) – url to the "Code Tables and Index (ZIP)" file
- **root_name** (*str, option*) – arbitrary name for the root of the hierarchy. Defaults to "root."
- **return_intermediates** (*bool*) – If True, return the untangle element and codes. Defaults to False.

Returns icd10 hierarchy and ICD-10-CM codes

Return type Tuple[nx.Graph, List[str]]

icdcodex.datacleaning.**build_icd10cm_hierarchy_from_zip**(*code_desc_zip_fp*, *code_table_zip_fp*, *root_name: Optional[str] = None*, *return_intermediates=False*)

build the icd10 hierarchy from zip files downloaded from cms.gov

Parameters

- **code_desc_zip_fp** (*Pathlike*) – file path to the "Code Descriptions in Tabular Order (ZIP)" file
- **code_table_zip_fp** (*[type]*) – file path to the "Code Tables and Index (ZIP)" file
- **root_name** (*str, option*) – arbitrary name for the root of the hierarchy. Defaults to "root."
- **return_intermediates** (*bool*) – If True, return the untangle element and codes. Defaults to False.

Returns icd10 hierarchy and ICD-10-CM codes

Return type Tuple[nx.Graph, List[str]]

icdcodex.datacleaning.**build_icd9_hierarchy**(*fp*, *root_name=None*)

build the icd9 hierarchy

Parameters

- **fp** (*Pathlike*) – Path to hierarchy spec, available at https://github.com/kshedden/icd9/blob/master/icd9/resources/icd9Hierarchy.json
- **root_name** (*str, option*) – arbitrary name for the root of the hierarchy. Defaults to "root."

Returns icd-9 hierarchy (nx.Graph) and ICD9 codes (List[str])

icdcodex.datacleaning.**build_icd9_hierarchy_from_url**(*url='https://github.com/kshedden/icd9/blob/master/icd9/r*, *root_name=None*)

build the icd9 hierarchy by downloading the hierarchy files

Parameters

- **url** (*str, optional*) – url to hierarchy spec. Defaults to "https://github.com/kshedden/icd9/blob/master/icd9/resources/icd9Hierarchy.json".
- **root_name** (*str, option*) – arbitrary name for the root of the hierarchy. Defaults to "root."

Returns icd-9 hierarchy (nx.Graph) and ICD9 codes (List[str])

icdcodex.datacleaning.**main**()

icdcodex.datacleaning.**traverse_diag**(*G*, *parent*, *untangle_elem*, *extensions=None*)
traverse the diagnosis subtrees, adding extensions as appropriate

Seventh-character extensions may be specified as a child, sibling or uncle/aunt. Also, some diagnoses are non-billable because they are, parents to more specific sub-diagnoses.

Parameters

- **G** (*nx.Graph*) – ICD hierarchy to mutate
- **parent** (*str*) – parent node
- **untangle_elem** (*untangle.Element*) – XML element, from untangle API
- **extensions** (*List[Tuple[str,str]], optional*) – Seventh character extensions and related descriptions. Defaults to None.

5.1.4 icdcodex.hierarchy module

deserialize icd hierarchies computed in datacleaning.py

icdcodex.hierarchy.**icd10cm**(*version: Optional[str] = None*) → Tuple[networkx.classes.graph.Graph, Sequence[str]]
deserialize icd-10-cm hierarchy

Parameters **version** (*str, optional*) – icd-10-cm version, including 2019 to 2020. If None, use the system year. Defaults to None.

Returns ICD-10-CM hierarchy and codes

Return type Tuple[nx.Graph, Sequence[str]]

icdcodex.hierarchy.**icd9**() → Tuple[networkx.classes.graph.Graph, Sequence[str]]
deserialize icd9 hierarchy

Returns ICD9 hierarchy and codes

Return type Tuple[nx.Graph, Sequence[str]]

5.1.5 icdcodex.icd2vec module

Build a vector embedding from a networkX representation of the ICD hierarchy

class icdcodex.icd2vec.**Icd2Vec**(*num_embedding_dimensions: int = 128*, *num_walks: int = 10*, *walk_length: int = 10*, *window: int = 4*, *workers=1*, ***kwargs*)
Bases: object

fit(*icd_hierarchy: networkx.classes.graph.Graph*, *icd_codes: Sequence[str]*, ***kwargs*)
construct vector embedding of all ICD codes

Parameters

- **icd_hierarchy** (*nx.Graph*) – Graph of ICD hierarchy

- **kwargs** – arguments passed to the Node2Vec.fit

to_code(*vecs: Union[Sequence[Sequence], numpy.ndarray]*) → Sequence[str]
decode continuous representation of ICD code(s) into the code itself

Parameters **vecs** (*Union[Sequence[Sequence], np.ndarray]*) – continuous representation of ICD code(s)

Returns ICD code(s)

Return type Sequence[str]

to_vec(*icd_codes: Sequence[str]*) → numpy.ndarray
encode ICD code(s) into a matrix of continuously-valued representations of shape m x n where m = self.num_embedding_dimensions and n = len(icd_codes)

Parameters **icd_codes** (*Sequence[str]*) – list of icd code(s)

Raises **ValueError** – If model is not fit beforehand

Returns continuously-valued representations if ICD codes

Return type np.ndarray

5.1.6 Module contents

CHAPTER

SIX

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/icdcodex/icdcodex/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

icdcodex could always use more documentation, whether as part of the official icdcodex docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jeremyadamsfisher/icdcodex/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *icdcodex* for local development.

1. Fork the *icdcodex* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/icdcodex.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv icdcodex
$ cd icdcodex/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 icdcodex tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/jeremyadamsfisher/icdcodex/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ pytest tests.test_icdcodex
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SEVEN

CHANGE LOG

CHAPTER

EIGHT

HISTORY

8.1 0.4.4 and 0.4.5 (2020-10-18)

- Add the code descriptions for ICD9
- Add usage on how to recapitulate functionality of sirrice/icd9
- Make the hierarchy directed to allow simpler and more intuitive traversal
- Fix issue where edges were not being formed between "Diseases Of The Blood And Blood-Forming Organs" and "Congenital Anomalies" and their children

8.2 0.4.3 (2020-10-04)

- Fix issue where hierarchy jsons were not being shipped with the pypi distribution

8.3 0.4.2 (2020-10-03)

- Add support for python <= 3.8 in the `hierarchy` module by using the `importlib.resources` backport

8.4 0.4.1 (2020-09-11)

- Update PyPI metadata

8.5 0.4.0 (2020-09-11)

- ICD-10-CM (2019 to 2020) codes are now fully present (whereas hackathon version missed certain codes)
- Versions of the ICD 9 and ICD-10-CM hierarchies are now cached to the `data` module
- Changed the hierarchy API: `hierarchy.icd9hierarchy()` is now `hierarchy.icd9()`. Ditto for ICD-10-CM.

8.6 0.3.0 (2020-09-05)

- Finesse API, now consistent between documentation and implementation

8.7 0.1.0 (2020-09-04)

- First release on PyPI, testing the waters during hackathon

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

INDEX