negbio Documentation

Release 2.0

Yifan Peng

Jan 02, 2021

The User Guide:

NegBio is a high-performance NLP tool for negation and uncertainty detection in radiology reports.

CHAPTER 1

Beloved Features

- Patterns on both universal dependency graph and regular expressions
- Creating user patterns
- Transparency
- Multiprocessing

NegBio officially supports Python>=3.6.

These instructions will get you a copy of the project up and run on your local machine for development and testing purposes. The package should successfully install on Linux (and possibly macOS).

1.1 Installation of NegBio

This part of the documentation covers the installation of NegBio. The first step to using any software package is getting it properly installed.

1.1.1 Prerequisites

- python >=3.6
- Linux
- Java

Note: since v1.0, MetaMap is not required. You can use the vocabularies (e.g., `patterns/cxr14_phrases_v2.yml`) instead.

1.1.2 Installation of MetaMap

If you want to use MetaMap to extract findings!!!

1. Download MetaMap full version and extract inot the directory called `public_mm`.
2. Install MetaMap locally. Installation instructions can be found at https://metamap.nlm.nih.gov/Installation.shtml.

```
cd public_mm
./bin/install.sh
```

3. Start the server.

```
./bin/skrmedpostctl start
./bin/wsdserverctl start
```

1.1.3 Getting the source code

NegBio is actively developed on GitHub, where the code is always available.

You can clone the public repository

```
$ git clone https://github.com/ncbi-nlp/NegBio.git
$ cd negbio
```

Once you have a copy of the source, you can prepare a virtual environment

```
$ conda create --name negbio python=3.6
$ source activate negbio
$ pip install --upgrade pip setuptools
```

or

```
$ virtualenv --python=/usr/bin/python3.6 negbio_env
$ source negbio_env/bin/activate
```

Finally, you can install the required packages:

```
$ pip install -r requirements3.txt
```

1.2 Quickstart

Eager to get started? This page gives a good introduction in how to get started with NegBio.

First, make sure that NegBio is installed.

1.2.1 Preparing the dataset

The inputs of NegBio should be in the BioC format.

Briefly, a BioC-format file is an XML document as the basis of the BioC data exchange and the BioC data classes. Each file contains a group of documents. Each document should have a unique id and one or more passages. Each passage should have (1) a non-overlapping offset that specifies the location of the passage with respect to the whole document, and (2) the original text of the passage.

The text can contains special characters such as newlines.

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<collection>
  <source>ChestXray-NIHCC</source>
  <date>2017-05-31</date>
  <key></key>
  <document>
    <id>0001</id>
    <passage>
      <offset>0</offset>
      <text>findings:
chest: four images:
right picc with tip within the upper svc.
probable enlargement of the main pulmonary artery.
mild cardiomegaly.
no evidence of focal infiltrate, effusion or pneumothorax.
dictating </text>
    </passage>
  </document>
  <document>
    <id>0002</id>
    <passage>
      <offset>0</offset>
      <text>findings: pa and lat cxr at 7:34 p.m.. heart and mediastinum are
stable. lungs are unchanged. air- filled cystic changes. no
pneumothorax. osseous structures unchanged scoliosis
impression: stable chest.
dictating </text>
    </passage>
  </document>
</collection>
```

1.2.2 Running NegBio

```
$ export OUTPUT_DIR=examples-local
$ export OUTPUT_LABELS=examples-local/labels.csv
$ export INPUT_FILES="examples/1.xml examples/2.xml"
$ bash examples/run_negbio_examples.sh
```

You can also include all reports in one folder, so that the `$INPUT_FILES=examples/*.xml`

After the script is finished, you can find the labels at `examples-local/labels.csv`. It contains three rows with respect to three documents. Each row has multiple findings, such as Atelectasis and Cardiomegaly. The definition of findings can be found at `patterns/cxr14_phrases_v2.yml`. In this file, 1 means positive findings, 0 means negative findings, and -1 means uncertain findings.

Besides the final label file, 6 folders contain the intermediate files of each step, respectively. For example, the `ssplit` folder consists of sentences, and the `parse` folder consists of the parse tree of each sentence. The content and format of these files should be self-explained.

Ready for more? Check out the `Advanced Usage` section.

1.3 Advanced Usage

This document covers some of NegBio more advanced features.

1.3.1 Running the pipeline step-by-step

The step-by-step pipeline generates all intermediate documents. You can easily rerun one step if it makes errors. The whole steps are

1. `text2bioc` combines text into a BioC XML file.
2. `normalize` removes noisy text such as `[**Patterns**]`.
3. `section_split` splits the report into sections
4. `ssplit` splits text into sentences.
5. Named entity recognition
    1. `dner_mm` detects UMLS concepts using MetaMap.
    2. `dner_regex` detects concepts using the vocabularies such as `patterns/cxr14_phrases_v2.yml`.
6. `parse` parses sentence using the Bllip parser.
7. `ptb2ud` converts the parse tree to universal dependencies using Stanford converter.
8. `neg2` detects negative and uncertain findings.
9. `cleanup` removes intermediate information.

1.3.2 General arguments

The general command is

```
python negbio/negbio_pipeline.py <command> [options] --output=/path/to/output/dir /
↪path/to/inputs
```

The `<command>` must be one of the steps above. The `--output` specifies the output directory. The `inputs` can be one or multiple files.

Other options include

1. `--suffix`: Append an additional `SUFFIX` to file names.
2. `--verbose`: Print more information about progress.
3. `--workers`: Number of threads.
4. `--files_per_worker`: Number of input files per worker.
5. `--overwrite`: Overwrite the output file.

1.3.3 Convert text files to BioC format

You can skip this step if the reports are already in the BioC format. **If you have lots of reports, it is recommended to put them into several BioC files, for example, 100 reports per BioC file.**

```
export BIOC_DIR=/path/to/bioc
export TEXT_DIR=/path/to/text
python negbio/negbio_pipeline.py text2bioc --output=$BIOC_DIR/test.xml $TEXT_DIR/*.txt
```

Another most commonly used command is:

```
find $TEXT_DIR -type f | python negbio/negbio_pipeline.py text2bioc  --output=$BIOC_
↪DIR
```

1.3.4 Normalize reports

This step removes the noisy text such as `[**Patterns**]` in the MIMIC-III reports.

1.3.5 Split each report into sections

This step splits the report into sections. The default section titles is at `patterns/section_titles.txt`. You can specify customized section titles using the option `--pattern=<file>`.

1.3.6 Splits each report into sentences

This step splits the report into sentences using the NLTK splitter (nltk.tokenize.sent_tokenize).

1.3.7 Named entity recognition

This step recognizes named entities (e.g., findings, diseases, devices) from the reports. In general, MetaMap is more comprehensive while vocabulary is more accurate on 14 types of findings. MetaMap is also slower and easier to break than vocabulary.

1.3.7.1 Using MetaMap

The first version of NegBio uses MetaMap to detect UMLS concepts. Please make sure that both `skrmedpostctl` and `wsdserverctl` are started

MetaMap intends to extract all UMLS concepts. Many of them are not irrelevant to radiology. Therefore, it is better to specify the UMLS concepts of interest via `--cuis=<file>`

```
$ export METAMAP_BIN=METAMAP_HOME/bin/metamap16
$ negbio_pipeline dner_mm --metamap=$METAMAP_BIN --output=$OUTPUT_DIR $INPUT_DIR/*.xml
```

1.3.7.2 Using vocabularies

NegBio also integrates the CheXpert's method to use vocabularies to recognize the presence of 14 observations. All vocabularies can be found at `patterns`. Each file in the folder represents one type of named entities with various text expressions. You can specify customized patterns via `--phrases_file=<file>`.

1.3.8 Parse the sentence

This step parses sentence using the Bllip parser.

1.3.9 Convert the parse tree to UD

This step converts the parse tree to universal dependencies using Stanford converter.

1.3.10 Detect negative and uncertain findings

This step detects negative and uncertain findings using patterns. By default, the program uses the negation and uncertainty patterns in the `patterns` folder. However, You can specify customized patterns such as `--neg-patterns=<file>`.

1.3.10.1 Patterns on the dependency graph

The pattern is a semgrex-type pattern for matching node in the dependency graph. Currently, we only support < and > operations. A detailed grammar specification (using PLY, Python Lex-Yacc) can be found in `ngrex/parser.py`.

Since v2.0, NegBio integrates the CheXpert algorithms. NegBio utilizes a 3-phase pipeline consisting of pre-negation uncertainty, negation, and post-negation uncertainty (Irvin et al., 2019). Each phase consists of rules which are matched against the mention; if a match is found, then the mention is classified accordingly (as uncertain in the first or third phase, and as negative in the second phase). If a mention is not matched in any of the phases, it is classified as positive.

You can specify customized patterns via `--neg-patterns=<file>`, `--pre-uncertainty-patterns=<file>`, and `--post-uncertainty-patterns=<file>`. Each file is an yaml-format file that consists of a list of patterns. Each pattern must have an `id` field and a `pattern` field. This allows NegBio to associate each pattern with the detected negation/uncertainty, to maximum the transparency. Examples can be found at `patterns`.

1.3.10.2 Regular expression patterns

NegBio also allows to use the regular expression to match simple cases. This function can also speed up the detection process, because pattern matching on the dependency graph is relatively slower. NegBio will first use regular expressions to match the text. If not found, semgrex is then used.

You can specify customized patterns via `--neg-regex-patterns=<file>` and `--uncertainty-regex-patterns=<file>`. Each file is an yaml-format file that consists of a list of patterns. Each pattern must have an `id` field and an `pattern` field. Examples can be found in `patterns`.

1.3.11 Cleans intermediate information

This step removes intermediate information (sentence annotations) from the BioC files.

1.4 NegBio Developer Guide

1.4.1 Create this documentation

```
$ pip install Sphinx sphinx_rtd_theme recommonmark
$ cd docs
$ make html
```

1.4.2 Testing the code

```
$ python -m pytest tests
```

1.5 License

Peng Y, Wang X, Lu L, Bagheri M, Summers RM, Lu Z. NegBio: a high-performance tool for negation and uncertainty detection in radiology reports. AMIA 2018 Informatics Summit. 2018, 188-196.

Wang X, Peng Y, Lu L, Bagheri M, Lu Z, Summers R. ChestX-ray8: Hospital-scale Chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017, 2097-2106.

1.6 Contributing

When contributing to this repository, please first discuss the change you wish to make via issue, email, or any other method with the owners of this repository before making a change.

This project adheres to the Contributor Covenant Code of Conduct.

1.7 Maintainers

NegBio is maintained with :heart: by:

– **@yfpeng**

See also the list of contributors who participated in this project.

1.8 Acknowledgments

This work was supported by the Intramural Research Programs of the National Institutes of Health, National Library of Medicine and Clinical Center.

We are grateful to the authors of NegEx, MetaMap, Stanford CoreNLP, Bllip parser, and CheXpert labeler for making their software tools publicly available.

We thank Dr. Alexis Allot for the helpful discussion.

1.9 Disclaimer

This tool shows the results of research conducted in the Computational Biology Branch, NCBI. The information produced on this website is not intended for direct diagnostic use or medical decision-making without review and oversight by a clinical professional. Individuals should not change their health behavior solely on the basis of information produced on this website. NIH does not independently verify the validity or utility of the information produced by this tool. If you have questions about the information produced on this website, please see a health care professional. More information about NCBI's disclaimer policy is available.

1.10 Reference

- Peng Y, Wang X, Lu L, Bagheri M, Summers RM, Lu Z. NegBio: a high-performance tool for negation and uncertainty detection in radiology reports. *AMIA 2018 Informatics Summit*. 2018, 188-196.
- Wang X, Peng Y, Lu L, Bagheri M, Lu Z, Summers R. ChestX-ray8: Hospital-scale Chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 2097-2106.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search