NLP for Annotation, Extraction and Classification of Pulmonary Embolism MIMIC III clinical notes

1. Introduction

Pulmonary embolism is the sudden blockage of a major blood vessel (artery) in the lung. In most cases, pulmonary embolism is caused by blood clots that travel to the lungs from the legs or other parts of the body. If the clot is large and stops blood flow to the lung, it can be deadly. Pulmonary embolism occurs between 300,000 and 600,000 cases, which results in between 50,000 and 200,000 deaths each year in the United States. However it is the most common preventable. Quick treatment could save life or reduce the risk of future problems.

Goal of the project:

Extract and annotate mentions and relations that closely related to pulmonary embolism. Classify the clinical documents into: Affirmed and Negated, and then Evaluate the system classified documents with manually classified ones.

2. Methods

SDLC: Planning:

The system uses 2 target concepts, d-dimer and pulmonary embolism (PE).

- D-dimer is a fibrin degradation product, a small protein fragment present in the blood after a blood clot is degraded by fibrinolysis. it contains two D fragments of the fibrin protein joined by a cross-link. D-dimer testing is of clinical use when there is a suspicion of deep venous thrombosis (DVT), pulmonary embolism (PE) or disseminated intravascular coagulation (DIC). Normal d-dimer (<= 500ng/mL) has almost 100% negative predictive for PE. If the d-dimer reads high, then further testing (CT pulmonary angiography, etc.) is required.
- CT pulmonary angiogram (CTPA) results are used to confirm or exclude a diagnosis of PE after first-line tests. The results have became the gold standard for diagnosis of pulmonary embolism.

SDLC: Analysis:

The steps of evaluation the feasibility of the system are include 1. the accessibility of the documents; 2. document selection; 3. training set and test set selection; 4. manual annotation; 5. Develop targets rules and modifier rules, and then feature inference rules and doc inference rules; 6. Design, debug and tune up the pipeline; 7. Test the system; 8. Evaluation.

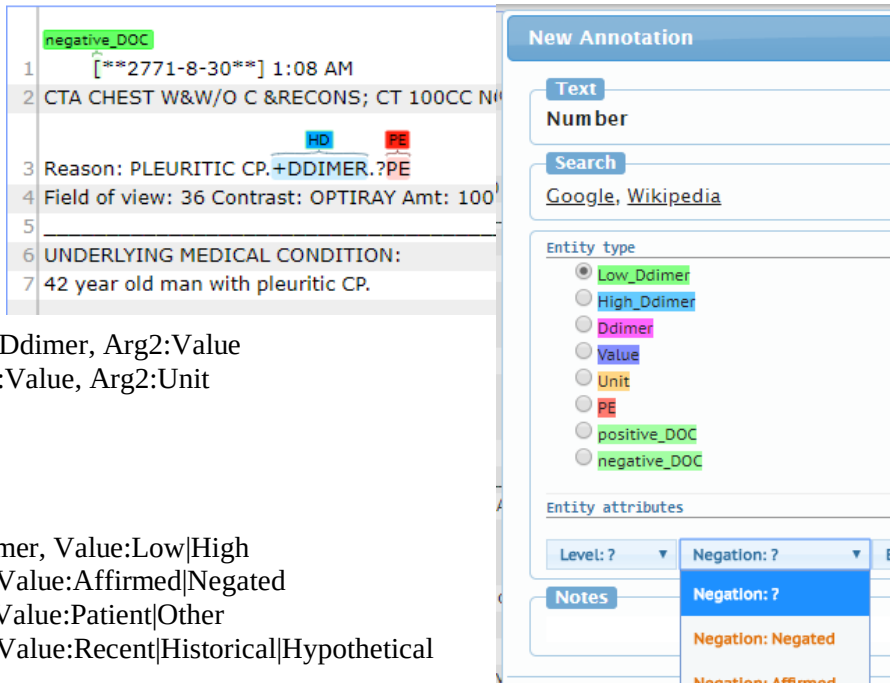
The documents we used MIMIC-III database. After we got the approval for access of MIMIC-III database from PhysioNetWorks, physionet.org, we downloaded the documents from MySQL server.

The system has 2 pipelines, d-dimer and PE, so d-dimer, pulmonary embolism, and CT (which is used to diagnose PE) were used in the database query. We did not ICD-9 CM code for the query, because the combination of d-dimer, pulmonary embolism, and CT is clear enough to define the documents. We found that in most of the documents, there are several d-dimer and pulmonary emboli/lus/lism mentions in the documents, so that the order of d-dimer and pulmonary emboli in the query should not have big effect for the selection of the documents. Following is the MySQL query:

```
SELECT subject_id, text  FROM NOTEEVENTS
WHERE text LIKE '%d-dimer% CT %pulmonary emboli%'
```

Totally 335 documents were selected and saved into BRAT directory.

For manual annotation of the reference standard, we defined 3 types of entities and related attributes for our annotation schema.



[entities]
Low_Ddimer
High_Ddimer
PE
pos_DOC
neg_DOC

[relations]
Concept_Value Arg1:Ddimer, Arg2:Value
Value_Unit Arg1:Value, Arg2:Unit

[events]

[attributes]
Level Arg:Ddimer, Value:Low|High
Negation Arg:PE, Value:Affirmed|Negated
Experiencer Arg:PE, Value:Patient|Other
Temporality Arg:PE, Value:Recent|Historical|Hypothetical

Beside "PE" for pulmonary embolism (PE), because d-dimer expressed in many forms in the clinical notes, some in name value unit format, others used different modifiers "positive", "+", "elevated", etc. to describe d-dimer, we decided to just use low d-dimer (LD) and high d-dimer (HD) instead of name value and unit annotations. We also included document level labels in our annotation schema, which is used for document level evaluation.

From 335 documents we saved, we used Python code to randomly select 30 documents and moved into training folder, and randomly selected 20 documents and moved into testing folder. Because we used random selection, the ratio of annotation types in training and testing sets should represent the whole set.

SDLC: System design and implementation
The system has 2 pipelines.

d-dimer pipeline:

we defined 3 d-dimer rules to try to match all types of d-dimer format in the documents and also avoid overlapping annotation.

```
rule=r'(?P<name>(d-dimer|ddimer))
        (?P<n1>.{1,25}?)
        (?P<value>[0-9]{1,4}(\.[0-9]{0,3})?\s*)
        (?P<n2>[^\n\w\d]*)
        (?P<unit>(ug\/l|ng\/ml|mg\/l|nmol\/l)?)'
rule1=r'(elevated|pos|positive|increased|high|\+)(.{1,20})?(\n)?\s?(d-dimer|d\s?dimer)'
rule2=r'(d-dimer|d\s?dimer)([^0-9-:]{1,15})?(positive|pos)'
```

Following is the pipeline:

- Instantiation pipeUtils Annotation and Document
- Read and pre-process the clinical notes
- Sequentially apply d-dimer rules, annotate mentions
- Save (append) annotations to each annotation files (same as BRAT ann file for consistency)
- Document classification
- Mention level evaluation
    - Read annotation file and convert to data frame
    - Convert data frame to annotation object
    - Mention level evaluation by compare with manually annotated BRAT files.

The module in d-dimer pipeline:

- pipeUtils.py module: to organize annotations via instants of Document and Annotation.

The functions of d-dimer pipeline:

- ddimer_val: process documents and return doc.annotations object.
- df2ann: convert annotation dataframe to annotation object.
- convert annotations to dataframe.
- compare2ann_types_by_span: mention level evaluation.

PE pipeline:

- Read rules
- Instantiation MyPipe (wrapper of PyContextNLP using PyRush as sentence segmentor). For our another PE pipeline (new pipeline), we used pipeUtils and modified PadClassificationSystem class.
- Document level evaluation
    - By compare with manually annotated BRAT files, which has document level marks inside
- Mention level evaluation
    - By compare with manually annotated BRAT files.

The module in PE pipeline:

- pynlp_pipe_pe.py: a wrapper of PyContextNLP using PyRush as the sentence segmentor. It return entity annotations, relationships and document classification.
- pynlp_valid.py: for evaluation.
- DocumentClassifier_pe.py: revised DocumentClassifier.py

The functions used in PE pipeline:

- Save annotation dataframes to annotation files (BRAT ann format)
- convert annotations to dataframe.
- compare2ann_types_by_span: mention level evaluation.

DocumentClassifier class in DocumentClassifier.py module has evaluation function, but we did not use. We encountered some problems, maybe because of the different system.

Following tools, resources were used in our system:

- PyRush.py
- pyContextNLP.py
- radnlp.py
- DocumentClassifier.py
- itemData.py
- visual.py

In one of our pe pipeline, we used DocumentClassifier.py module. For easy match NLP system annotation spans with BRAT annotation spans, we used pipeUtils.py and modified PadClassificationSystem class instead.

SDLC: Testing

The steps to evaluate the efficacy include:

- Iteratively load and process all documents by pynlp_pipe_pe module, and save the results in a Python dictionary with document name as the key and classified result as the value.
- Iteratively read BRAT ann files and check the document level labeling, and save the results in a another dictionary with document name as the key and classified result as the value.
- Compare these 2 dictionary and print confusion Matrix, and calculate precision, recall and F1 values.

The results of our testing set were showed as following:

	BRAT	Reference	Total
NLP	8	0	8
System	2	10	12
Total	10	10	

Precision: 1.0; Recall: 0.8; F1: 0.89

We saved annotations to annotation files and keeps the same format as BRAT annotation files. The columns including: “markup_id”, “type”, “start”, “end”, and “txt”. They are saved as a plain text file, so the data type of the columns are all converted to string. We have 20 documents in our test set. By chance, there are 10 positive and 10 negative notes.

Here, one false negative is caused by 2 different modifiers in one sentence, i.e. “No central pulmonary embolus is identified on the left, …. Subsegmental pe involving the right…”. Another false negative is caused by sentence segment reason.

Error analysis

We used mention level evaluation to analyze the errors in fp and fn documents during our training process, which including:

- Read ann file of selected fp or fn document
- Convert annotation to dataframe
- Convert dataframe to Annotation instance
- Compare the annotation
- According to the annotation id (ann_id) to find the errors.

SDLC: Deployment

The NLP system was deployed to run the rest of PE MIMIC-III files we downloaded from MySQL server. The results were save to output.csv file. A total of 1928 lines in the file. Depend on the notes, there are about 7 records for each document.

Our 2 revised pipeline (one for PE and one for d-dimer) were also deployed to run the whole 335 PE MIMIC-III documents. The mention level results were save to out_table.csv and out_table_dimer.csv files respectively. There are 1964 lines for PE pipeline and 572 lines for d-dimer pipeline. The following are some examples:

From PE pipeline:

record_id,subject_id,note_id,annotation_type,span_start,span_end,PAD_snippet,neg_flag
83596_0_1,83596,83596_0,high_ddimer,1947,1961,"D-Dimer 5515, ",0
83596_0_2,83596,83596_0,high_ddimer,4847,4860,D-DIMER-5515*,0

From d-dimer pipeline:

record_id,subject_id,note_id,annotation_type,span_start,span_end,PAD_snippet,neg_flag
83596_0_1,83596,83596_0,pe_ann,1982,1984,PE,0
83596_0_2,83596,83596_0,pe_ann,16910,16912,PE,0
83596_0_3,83596,83596_0,pe_ann,16839,16857,Pulmonary Embolism,0

3. Discussion

From this class and this project, we learned a lot about the principles of rule-based NLP, and the use of regular expression, sentence segmentor, pyContextNLP, pipeUtils, wrapperPyConText, etc. We can use these tools to build our simple rule-based NLP system.

Our system only classify the documents into negated and affirmed (positive and negative). At this time, our system has two pipelines, so can only classify 2 concepts, d-dimer and PE. Because PE may occur either on left or right site, our system has problem to classify sentences such as "… no pe on left … but on right …".

The lessons we learned from the process include:

- For regular expression rules, try to avoid overlap between rules, which will cause repeat annotate some mentions.
- Try to use less greedy rules. Some regular expression will match more than we want that may cause problems.
- Try to use well restricted target rules and modifier rules, and avoid broad rules.
- Define suitable context window is important.

Published article that described an NLP system that targeted the same concept is Dr. Chapman, et al's paper. They used peFinder, a pyContextNLP based CTPA report classifier to classify the documents and compared with two bag-of-word naive Bayesian classifiers. Their data set is from retrospectively identified 4,657 CTPA reports that were ordered to rule out pulmonary emboli. They obtained a very good results.

We used MIMIC-III documents instead of CTPA reports and used 20 documents for testing.

Our impression about the project is that we learned a lot from this class, and we can build rule-based NLP pipeline by different methods and conduct extraction, annotation and classification work. Using Document object and Annotation object are very convenience for NLP process. Careful develop target rules and modifier rules are very important.

4. References

Chapman BE, Lee S, Kang HP, Chapman WW. Document-level classification of CT pulmonary angiography reports based on an extension of the ConText algorithm. J Biomed Inform. 2011 Oct;44(5):728-37. doi: 10.1016/j.jbi.2011.03.011.

Chapman BE. pyConTextNLP. Available from: https://github.com/chapmanbe/pyConTextNLP.