Annotating and Detecting Medical Events in Clinical Notes

Prescott Klassen, Fei Xia, and Meliha Yetisgen
University of Washington
PO Box 352425, Seattle, WA 98195, USA
{klassp, fxia, melihay}@uw.edu

Abstract

Early detection and treatment of diseases that onset after a patient is admitted to a hospital, such as pneumonia, is critical to improving and reducing costs in healthcare. Previous studies (Tepper et al., 2013) showed that change-of-state events in clinical notes could be important cues for phenotype detection. In this paper, we extend the annotation schema proposed in (Klassen et al., 2014) to mark change-of-state events, diagnosis events, coordination, and negation. After we have completed the annotation, we build NLP systems to automatically identify named entities and medical events, which yield an f-score of 94.7% and 91.8%, respectively.

Keywords: Annotation, Event Extraction, Named-entity recognition

1. Introduction

Clinical notes describe the progress of disease in great level of detail. Although notes individually present a single snapshot in time, the clinical narrative in each note often makes explicit or implicit references to a previous note. To capture this level of information is important for the detection of phenotypes/diseases such as pneumonia. For instance, in reports that do not explicitly mention that the patient has pneumonia, an important cue for pneumonia detection is pneumonia-related symptoms and their change-of-state through time, which is often encoded in sequential ICU progress notes and chest x-ray reports. As an example, the text snippet "*No change in diffuse lung disease consistent with edema. Patchy bilateral lung consolidation has diminished. No new focal abnormalities*" belongs to a chest x-ray note of a patient who is labeled in the gold standard as *negative for pneumonia*. Our pneumonia prediction system mislabels the patient as *positive for pneumonia*, because the system primarily used n-grams and UMLS concepts as features and did not capture change-of-state information in the feature representation (Tepper et al., 2013).

To address this problem, in our previous study (Klassen et al., 2014), we proposed to annotate change-of-state events in the text snippets with a tuple schema of five fields ⟨location, attribute, value, change-of-state, reference⟩. In this paper, we discuss the extensions we have made to the schema to add annotation of diagnosis events and to handle coordination and negation of tuple fields. We then report on the experimental results of the NLP systems we build to automatically identify change-of-state and diagnosis events.

2. Creating a Corpus for Phenotype Detection

Early detection and treatment of ventilator associated pneumonia (VAP) is important as it is the most common healthcare-associated infection in critically ill patients. Even short-term delays in appropriate treatment for patients with VAP are associated with high mortality rates, longer-term mechanical ventilation, and excessive hospital costs. Our research goal is to build NLP systems which assist healthcare practitioners in identifying patients who are developing critical illnesses (e.g., VAP).

```
01 CHEST, PORTABLE 1 VIEW
02 INDICATION:
03 Shortness of breath
04 COMPARISON: July 16 10 recent prior
05 FINDINGS:
06 Left central line, tip at mid-SVC.
07 Cardiac and mediastinal contours as before
08 No pneumothorax.
09 Lungs: Interval increase in right lung base
10 pulmonary opacity with air bronchograms,
11 increasing  pneumonitis / atelectasis.
```

Figure 1: A sample chest x-ray report

2.1. PNA/CPIS Detection

In our previous study (Tepper et al., 2013), we created an annotated corpus of 1344 chest x-rays from the UW Harborview Medical Center. Annotators read each report and determined whether the patient had pneumonia (PNA) and recorded a clinical pulmonary infection score (CPIS) (Zilberberg and Shorr, 2010). CPIS is used to predict which patients will benefit from the invasive, and preferably avoidable procedure to obtain pulmonary cultures. There are three labels for CPIS: *1A (no infiltrate)*, *1B (diffuse infiltrate or atelectasis)*, and *1C (local infiltrate)*. Similarly, there are three labels for PNA: *2A (no suspicion of PNA)*, 2B (suspicion of PNA), and *2C (probable PNA)*. In addition to labels for CPIS and PNA, we also asked the annotators to highlight the text snippet in the chest x-rays that supports the labels that annotators choose for the x-rays. We call these snippets *rationale snippets* (see (Yu et al., 2011) for a similar approach). Figure 1 shows a sample x-ray report and Line 9-11 is marked as a rationale snippet for CPIS/PNA. Based on this corpus we built an NLP system and achieved classification accuracy 87.1% for CPIS and 82.1% for PNA (Tepper et al., 2013).

Error analysis for both classification tasks revealed that reducing classification errors required features that go beyond simple word ngrams, which motivated the annotation

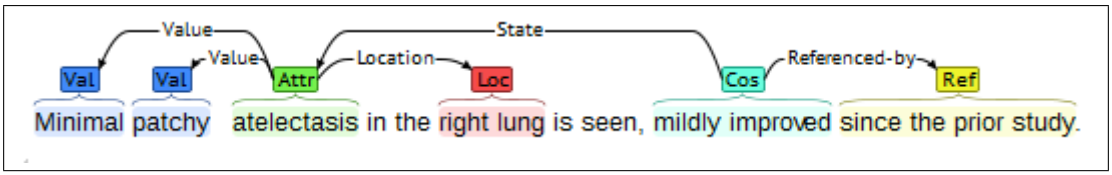

Figure 2: A snippet featuring the annotation of a change-of-state (cos) event

of change-of-state described below.

2.2. Annotating change of state

To annotate change-of-state (cos) event, we define a cos event as a tuple ⟨cos, attr, loc, val, ref⟩ (an example is given in Fig 2): loc is the anatomical location (e.g., *right lung* in Fig 2), attr is an attribute of the location that the event is about (e.g., *atelectasis*), val is a possible value for the attribute (e.g., *minimal patchy*), cos indicates the change of state for the attribute value compared to some previous reports (e.g., *mildly improved*), and ref is a link to the report(s) that the change of state is compared to (e.g., *since the prior study*). Not all tuples will have values for all five fields. A field can be unspecified and inferred from the context of the surrounding snippet text or from the collection of snippets that have been extracted from the sequence of a patient's x-ray reports.

As a sentence can contain multiple events, we need to group the fields of an event together. To achieve that, we use directed, labeled arcs to link two fields. There are four arc labels:

1. Label *State* connects a cos entity with an attr or a loc entity
2. Label *Location* connects an attr entity with a loc entity
3. Label *Value* connects an attr or loc entity with a val entity
4. Label *ReferencedBy* connects a cos or an attr entity with a ref entity

We define a total order between fields, cos $\prec$ attr $\prec$ loc $\prec$ val $\prec$ ref and require that, in an arc $A \rightarrow B$, A must precede B according to the total order. As a result, the annotation of an event is a directed acyclic graph (DAG), as shown in Fig 2. For annotation, we use a lightweight, web browser-based annotation tool called BRAT (Stenetorp et al., 2012).[1] In the next section, we describe a few extensions to the annotation schema, and Table 1 shows the statistics of the corpus before and after the extensions.

3. Extending the Annotation Schema

We have made two main extensions to the annotation schema, as explained in this section.

[1] http://brat.nlplab.org/

	Before extensions	After extensions
X-ray reports	1344	
Unique snippets	1008	
Entity types	5	9
Arc label types	4	12
Entity tokens	7173	8474
Arc label tokens	4128	6329

Table 1: The corpus statistics before and after the extensions to the annotation schema

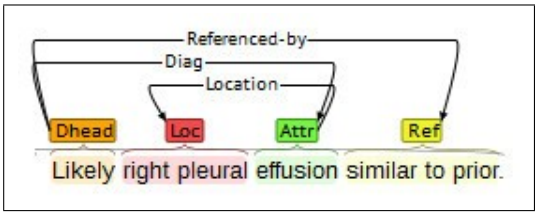

Figure 3: Annotation of a diagnosis event

3.1. Adding diagnosis annotation

The snippets often include a diagnosis statement (e.g., *likely right pleural effusion*), typically indicating a possible or likely diagnosis of a previously mentioned change of state. Because such a statement can be an important cue for phenotype detection, we extend our annotation schema to mark it with a new event type called *diagnosis event*. The diagnosis event is very similar to a cos event except that the cos field is replaced by a new field called dhead which is the head of a diagnosis statement (e.g., *likely* in *likely right pleural effusion*). In addition to dhead, the tuple for a diagnosis event may include the four fields (i.e., attr, loc, val, and ref); an example is given in Fig 3. We also need a new arc label, *Diag*, that connects dhead with attr.

3.2. Handling coordination and negation

Coordination and negation are common in snippets. In Fig 4, the three attr fields are connected by conjunction words *or* and *and*; the cos field *change* is negated by the word *No*.

To annotate coordination, we mark the conjunction (such as *and*, *or*, and *but*) as conj and words such as *versus* and *vs.* as versus. We distinguish five types of coordination, as illustrated in Fig 5, and use arc labels such as *combine* and *exclude* to indicate different relations between the conjuncts. Distinguishing the type of coordination could be important for phenotype detection; for instance, coordination with multiple *Alternate* arcs could indicate hedging in

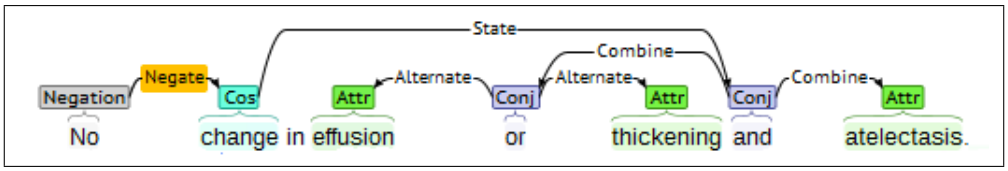

Figure 4: An example that includes both coordination and negation

a diagnosis event.

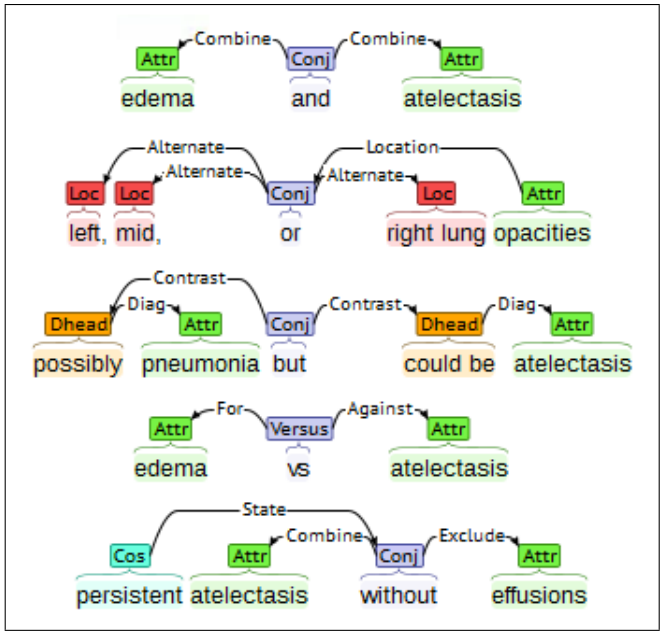

Figure 5: Five types of Coordination

Similarly, we use the arc label *Negate* to link a negation word (`negation`) with the word it negates. One nice property of this annotation schema is that the structure for an event is guaranteed to be a tree because each tuple field has at most one parent and there are no loops due to the total order between fields. Consequently, we can treat the event detection task as a dependency parsing problem, as discussed in Section 4.2.

In summary, the new annotation schema marks nine entity types: six as fields in a change-of-state or a diagnosis event (`cos`, `attr`, `loc`, `val`, `ref`, and `dhead`), two for coordination (`conj` and `versus`), and one for negation (`negation` for negation words). The arc label set has 12 members: five for arcs connecting two event fields (*State*, *Location*, *Value*, *ReferencedBy*, and *Diag*), six for arcs connecting a conjunction and a conjunct (*Combine*, *Alternate*, *Contrast*, *Exclude*, *For*, and *Against*), and one for arcs connecting a negation word and a field (*Negate*). The numbers of entities and arcs in our corpus are provided in Tables 2 and 3, respectively.

4. Building an Event Detection System

Our updated annotation schema was applied by a single annotator to all 1008 unique snippets in the rationale snippet corpus. Then we use this corpus to train and evaluate an event extraction system. We treat event extraction as a dependency parsing task (McClosky et al., 2011), and the process has two stages: first, it identifies named entities in the text snippets with a named entity recognizer; second, it assigns labeled arcs between entities and forms event trees by running a dependency parser. For both named-entity recognition and dependency parsing, we train off-the-shelf tools with our corpus, and evaluate them using 5-fold cross validation.

Entity type	# of entities	P	R	F1
`attr`	2467	97.3	97.6	97.4
`cos`	1126	94.4	94.6	94.5
`dhead`	566	89.4	88.5	89.1
`loc`	2099	94.8	95.2	95.0
`ref`	196	89.7	88.2	89.0
`val`	1006	92.4	89.4	90.9
`conj`	711	93.9	94.8	94.4
`versus`	40	97.1	87.2	91.9
`negation`	263	96.1	96.8	96.5
Total/ Micro-avg	8474	94.8	94.6	94.7

Table 2: Performance of the Stanford Named Entity Recognizer. The first column lists the numbers of entities in the gold standard. The last three columns show system performance measured by precision (P), recall (R), and f-score (F1) when evaluated with 5-fold cross validation.

4.1. Named-entity recognition

For named-entity recognition, we use version 3.5.2 of the CRF-based Stanford Named Entity Recognizer (NER)[2] (Finkel et al., 2005), configured with the default settings[3]. For evaluation, we use the evaluation methods integrated into the Stanford NER package, and randomly divide our corpus into five folds for cross-validation.

The last three columns in Table 2 summarize the results of the NER experiments. For the most part, the system performed well with an average f-score of 94.7 across all entity types. The lowest performing entities, `dhead` and `ref`, were the ones in the set that were most likely to contain general and productive non-domain specific language, such as terms of likelihood and possibility in the case of `dhead`, and general descriptions of previous reports, conditions, or patient state in the case of `ref`.

Arc label	System	Gold	P	R	F1
Diag	500	524	93.6	89.3	91.4
Location	1746	1816	95.7	92.0	93.8
ReferencedBy	157	166	98.7	93.4	96.0
State	995	1066	92.0	85.8	88.8
Value	984	986	97.7	97.5	97.6
Alternate	433	430	94.5	95.1	94.8
Against	38	38	86.8	86.8	86.8
Combine	883	984	88.3	79.3	83.6
Contrast	9	17	11.1	5.9	7.7
Exclude	6	11	100.0	54.6	70.6
For	40	39	85.0	87.2	86.1
Negate	253	252	97.6	98.0	97.8
Subtotal/ Micro-avg	6044	6329	94.0	89.7	91.8
ROOT	2022	1743	82.7	96.0	88.9
DEP	2578	2572	98.1	98.4	98.3
Total/ Micro-avg	10644	10644	92.8	92.8	92.8

Table 3: Performance of dependency parsing by arc type. The first two columns show the numbers of dependency arcs in the system output and the gold standard, respectively. The last three columns show system performance measured by Precision (P), Recall (R), and F-Score (F1) when evaluated with 5-fold cross-validation.

4.2. Dependency parsing

The second stage of our event extraction process relies on dependency parsing to extract event trees from labeled text spans. We convert our change-of-state and diagnosis event annotations from the BRAT standoff format into the CoNLL format,[4] which was originally created for the CoNLL 2007 shared task on dependency parsing. For the conversion, we make non-final words of an entity depend on the final word in the entity with a dummy arc label *DEP*. Furthermore, the head of the entity at the root of an event tree will depend on a dummy node with arc label *ROOT*. For instance, given a sentence *w1 w2 w3 w4 w5*, let us assume that *w1 w2* and *w4 w5* are two entities and the first entity depends on the second entity with the label *rel*. After the conversion, *w1* will depend on *w2* with a *DEP* label, and the same is true for *w4* and *w5*; *w2* will depend on *w5* with label *rel* and *w5* on a dummy word with label *ROOT*; *w3* is ignored since it is not part of any entity. To train and evaluate the dependency parser, we use the named entities from the gold standard.

Notice that a rationale snippet can contain multiple sentences and a sentence can in turn contain multiple events. In our snippet corpus, there are 1008 snippets, 1268 sentences, and 1743 *ROOT arcs* (one for each event tree). For parsing, we used the Malt Parser[5](Nivre et al., 2007) version 1.8.1, and ran experiments with default parser settings. For evaluation, we used 5-fold cross validation and ran the evaluation tool MaltEval[6] (Nilsson and Nivre, 2008).

Arcs included for evaluation	LA	UAS	LAS
Without ROOT/DEP arcs	94.0	93.3	92.1
With ROOT/DEP arcs	92.8	92.3	91.6

Table 4: Performance of dependency parsing, measured by labeled accuracy (LA), unlabeled attached score (UAS), and labeled attachment score (LAS).

Table 3 shows the performance of the dependency parser by arc type. The first two columns list the numbers of dependency arcs in the system output and in the gold standard. The last three columns report parsing results measured by labeled precision, recall, and f-score. The top thirteen rows show the results for the twelve arc labels and the micro-average. The next two rows are the results for two dummy labels, *ROOT* and *DEP*, and finally the last row is the micro-average for all fourteen arc labels. The system performs well for all the labels except for *Exclude* and *Contrast* due to lack of training data.

Table 4 measures parsing performance in labeled accuracy (LA), which measures the correct dependency label, unlabeled attachment score (UAS), which measures the correctly identified head, and labeled attachment score (LAS), which measures the correctness of both the dependency label and the head. The first row excludes *ROOT* and *DEP* arcs in evaluation and the second row includes those arcs. The scores for the second row are lower mainly because identifying *ROOT* arcs is harder as a sentence may contain multiple event trees and each tree has a *ROOT* arc.

5. Conclusion and future work

Medical events in clinical notes are important cues for phenotype detection. In this study, we extended our change-of-state event annotation schema by adding annotation of diagnosis events, coordination, and negation. We then transformed our event tuples into event trees, which allowed us to treat event detection as a dependency parsing task. We trained an off-the-shelf named entity recognizer and a dependency parser on the annotated corpus, and the systems yielded an f-score of 94.7% and 91.8%, respectively. The corpus is available at http://depts.washington.edu/bionlp/datasets.htm.

For future work, we will first examine the effect of named entity recognition errors on the event extraction. We will then add event-based features to improve the performance of our phenotype detection system. Our ultimate goal is to use event detection, phenotype detection, and other NLP systems to monitor patients' medical conditions over time and prompt physicians with early warning, and thus improve patient healthcare quality while reducing the cost of healthcare.

References

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extrac-

[2]http://nlp.stanford.edu/software/CRF-NER.shtml

[3]http://nlp.stanford.edu/software/crf-faq.shtml#a

[4]http://nextens.uvt.nl/depparse-wiki/DataFormat

[5]http://www.maltparser.org

[6]http://www.maltparser.org/malteval.html

tion systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005)*, pages 363–370. Association for Computational Linguistics.

Klassen, P., Xia, F., Vanderwende, L., and Yetisgen, M. (2014). Annotating clinical events in text snippets for phenotype detection. In *Proceedings of LREC 2014*, Reykjavik, Iceland.

McClosky, D., Surdeanu, M., and Manning, C. D. (2011). Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT 2011)*, pages 1626–1635. Association for Computational Linguistics.

Nilsson, J. and Nivre, J. (2008). Malteval: an evaluation and visualization tool for dependency parsing. In *Proceedings of LREC*.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T.-k., Ananiadou, S., and Tsujii, J. (2012). Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*.

Tepper, M., Evans, H. L., Xia, F., and Yetisgen-Yildiz, M. (2013). Modeling annotator rationales with application to pneumonia classification. In *Expanding the Boundaries of Health Informations Using AI Workshop of AAAI 2013*.

Yu, S., Farooq, F., Krishnapuram, B., and Rao, B.-r. (2011). Leveraging rich annotations to improve learning of medical concepts from clinical free text. In *ICML workshop on Learning from Unstructured Clinical Text*, Bellevue, WA.

Zilberberg, M. D. and Shorr, A. F. (2010). Ventilator-associated pneumonia: the clinical pulmonary infection score as a surrogate for diagnostics and outcome. *Clinical Infectious Diseases*, 51(Suppl 1):S131–S135.