DoTAT: A Domain-oriented Text Annotation Tool

Anonymous ACL submission

Abstract

We propose DoTAT, a domain-oriented text annotation tool. The tool designs and implements functions heavily in need in domain-oriented information extraction. Firstly, the tool supports a multi-person collaborative process with automatically merging and review, which can greatly improve the annotation accuracy. Secondly, the tool provides annotation of event, nested event and nested entity, which are frequently required in domain-related text structuring tasks. Finally, DoTAT provides visualized annotation specification definition, automatic batch annotation and iterative annotation to improve annotation efficiency. Experiments on the ACE2005 dataset show that DoTAT can reduce the event annotation time by 19.7% compared with existing annotation tools. The accuracy without review is 84.09%, 1.35% higher than Brat and 2.59% higher than Webanno. The accuracy of DoTAT even reaches 93.76% with review. The demonstration video can be accessed from https://ecust-nlp-docker.oss-cn-shanghai.aliyuncs.com/dotat_demo.mp4. A live demo website is available at https://github.com/FXLP/MarkTool.

1 Introduction

A high-quality corpus is a prerequisite in supervised machine learning, especially for neural network-based model. Currently more and more domain-oriented information extraction tasks are proposed, therefore annotation tools should be redesigned to meet the requirements.

• Multiple specifications support There are many document types in each domain, and the schema of the target structured data are different. Therefore normally different annotation specification is defined for each document type. For example, in the medical domain, document types many include discharge summary, admission record, examination report, and operation record. In the public security domain, document types include fraud, theft, robbery, disputes and so on.

• Nested event Domain-oriented information extraction tasks often require event and nested event annotation. As shown in Figure 1, two separate sub-events (“victim” and “suspect”) are nested in the top “fraud” event. Traditionally event is defined as n-tuples and the trigger word is a verb, such as fraud event in Figure 1. In this paper, we take n-tuples of all forms as events, the trigger word can be a noun as a subject, and the arguments may be the attributes of the subject. For example, subject “victim” has multiple attribute-value pairs in Figure 1.

• Multi-person support with merging and reviewing Single-person annotation often leads to missing and wrong annotation due to human errors, the ambiguity of the words, or particular language phenomenon not covered by the specifications. In later experiment in Section 5.2, the accuracy may be less than 60% for new annotators. When there are multiple annotation specifications in domain-oriented
annotation tasks, more errors may appear since specifications vary and more annotators are required. Therefore, Multi-person collaborative annotation is required to improve the annotation quality. Furthermore the divergence between multiple annotators should be detected and the improved result can be achieved by automatically merging and human reviewing.

Among the existing text annotation tools, only Brat (Stenetorp et al., 2012) and Webanno (Yimam et al., 2013) support event annotation. However, the two do not design event annotation as a core function and do not contain enough features for specification management and quality improvement. To address the challenges above, we propose DoTAT, a domain-oriented text annotation tool for complex event annotation tasks. Besides the ordinary function such as visualized entity and relation annotation, its main features are as follows:

- **Visualized annotation specifications definition** The annotation specifications are defined by a visual interface instead of manual configuration so that administrators can easily define multiple specifications and annotators can dynamically select the specification to match their documents.

- **Merge and review** It provides pairwise consistency checking and automatic merging of content annotated by multiple people. The reviewer can also manually edit the merged content.

- **Iterative annotation** Annotators can re-load previous exported result file for further annotation. The function is frequently used in the situation that new version of a domain specification is designed and existing annotation file should be reused and revised. The above three features forms the basis of DoTAT annotation process and help to improve the quality of the annotation.

- **Nested event and nested entity** The tool not only supports nested event but also supports nested entity. Nested Entity means that one entity is inside another entity.

- **Automatic batch annotation** The tool provides automatic batch annotation by text matching based on regular expressions and dictionaries.

In the following section, we summarize annotation tools. Section 3 describes the overall workflow of DoTAT and its functions. Section 4 introduces the implementation of DoTAT. Section 5 illustrates the comparative experiment. Section 6 shows the case study in the medical and public security domains. Section 7 concludes this paper and gives further directions.

2 Related Work

There are various text annotation tools for different scenarios, but most of them do not support event annotation, including Knowtator (Ogren, 2006), WordFreak (Morton and LaCivita, 2003), Anafora (Chen and Styler, 2013), Atomic (Druskat et al., 2014), GATE Teamware (Bontcheva et al., 2013), Doccano and YEDDA (Yang et al., 2018).

Each tool has its own special features, e.g., WordFreak supports constituent parse structure and dependent annotations as well as ACE named-entity and coreference annotation. Doccano and YEDDA support the use of shortcut keys for entity annotation, and YEDDA can perform batch annotation through the command line.

Currently only Brat (Stenetorp et al., 2012) and Webanno (Yimam et al., 2013) support event annotation. However, it is difficult for them to annotate nested event. The method used by Brat and WebAnno for event annotation is to connect multiple entities through directed arcs. If the number of entities is numerous or the distance between entities is far, abundant arcs and intersections will appear on the whole page, resulting in an inferior visualization effect. Except for WordFreak, Anafora and Atomic, most tools declare to support multi-person collaborative annotation. GATE Teamware provides the adjudication interfaces to compare annotations. However, only Webanno provides the curation with automatic merging function. Compared to these tools, event annotation in
3 DoTAT Features Description

DoTAT is a web-based multilingual text annotation tool. There are three types of user roles: administrator, annotator, and reviewer. The fundamental annotation types include entity annotation, relation annotation, event annotation, and text classification. As shown in Figure 2, a typical annotation process using DoTAT may include the following five steps:

- **Define annotation specifications**: The administrator selects the annotation type and visually defines event types, entity types, relation types or text categories in annotation specifications.

- **Create and assign tasks**: Administrator creates and assigns tasks. Each task contains an annotation specification and several raw texts. It is recommended that two annotators and one reviewer are assigned to each task.

- **Annotate**: Before the annotators interactively annotate events or entities, they can use automatic batch annotation to accelerate the speed.

- **Merge and Review**: The reviewer starts consistency checking and automatic merging of the annotated content by multiple annotators. The reviewer can visually analyze the errors according to the merged events list. When there are too many similar errors, the reviewer can give feedback for administrator to redefine the annotation specification. With iterative annotation function, all existing annotations can be reused.

- **Export results**: After the review process, the annotated content can be exported by administrator to a result file and saved in JSON format.

### 3.1 Event Annotation

The annotation interface of DoTAT contains annotation panel and event list panel, as shown in Figure 3. Users can interactively annotate in the former panel, and the results are summarized in the later one. Users can also select an event in the event list panel and view this event in another panel.

When a user begins annotation, he can use dictionary matching or regular expression matching to automatically annotate entities to reduce manual efforts. For example, in the scenario of Figure 3, we use a medical organs dictionary to automatically annotate “lung” in the text. Then the user begins annotate events. He firstly selects the event type, then uses the mouse to pick a text span in the annotation panel, and then all arguments of this event type will appear immediately, the user can select an argument to annotate. As shown in Figure 3, the
annotator selects the argument “身体结构(body structure)” to annotate. The user repeatedly selects each span and corresponding argument to finish the event annotation. For the nested events, when the key of one event becomes an argument of another event, DoTAT considers the former as the internal event of the latter one. As shown in Figure 3, the key argument “lung” of the body structure event (7531) is nested in the event (7532) as an argument.

For the nested entity annotation, theoretically the internal entity overlaps the outer entity. In order to make both entities displayed well, we make the shadow of the internal entity a little smaller and put it in the top layer, the effects is shown on bottom left of Figure 3.

Algorithm 1 Automatically merge event annotations by using the Kuhn-Munkres Algorithm.

Input: \(A_n\): the n events of annotator-A; \(B_m\): the m events of annotator-B

Output: \(C\): the set of merged events; \(K\): the consistency checking score

1. Calculating the similarity matrix \(S_{n,m}\) of \(A_n\) and \(B_m\). Let \(S_{i,j}\) denote the element in the i-th row and j-th column of the matrix \(S_{n,m}\), and its value represents the similarity between the event \(a_i\) of \(A_n\) and the event \(b_j\) of \(B_m\).

2. Using Kuhn-Munkres Algorithm to find the optimal event merging strategy \(W_n\) in matrix \(S_{n,m}\).

3. for each event \(a_i\) in \(A_n\) do
   4. if \(a_i \in W_n\) then
      5. merge \(W_i\)
      6. if the original entities in the two events are the same then the state is ‘Consistent’
      7. else the state is ‘Inconsistent’
      8. end if
      9. add the merged event to the set \(C\)
     10. else
     11. add \(a_i\) to the set \(C\) with state ‘Only A’
     12. end if

13. end for

14. for each event \(b_j\) in \(B_m\) do
   15. if \(b_j \notin W_n\) then
      16. add \(b_j\) to the set \(C\) with state ‘Only B’
   17. end if
18. end for

19. \(K = \sum s_{i,j} / n\), where \(a_i \in W_n\) and \(b_j \in W_n\)

20. return \(C,K\).

3.2 Review of Event Annotation

The review procedure supports consistency checking, automatic merging, and manual revision. Before the review, the system will check the consistency of the annotated content of the two annotators. The problem is to find matched events between two annotated text, the detail is shown in Algorithm 1.

1) We calculate the similarity between each event. The event similarity is calculated as the number of matched entities divided by the number of all entities. The result is recorded as matrix \(S_{n,m}\). 2) Then the problem is defined as the maximum weight matching of weighted bipartite graphs. We apply the Kuhn-Munkres Algorithm to find optimized matching pairs. The consistency checking score is the sum of similarity values of matched pairs divided by the maximum number of events. When consistency checking score reaches the threshold, the system can start the merging process.

3) The merge criteria depends on the state, and there are four states for each event, “Consistent”, “Only A”, “Only B” and “Inconsistent”. The system automatically merges all the arguments for events in “Inconsistent” state. For the other three states, the system will only keep the larger event.

In the review procedure, the reviewer can view the merged annotations. If the reviewer doubts on the merged event, he can trace the source to view the original annotated event by clicking role switching bar to change current view. The reviewer can also perform manual modification. Typically they should modify the the events in “Inconsistent” state. The whole annotation process finishes after the reviewer submit the refined result.

4 Implementation

DoTAT is a web-based text annotation tool with the software license Apache-2.0. We uses the Vue.js and Element UI to build the user interface. The core of Vue.js is a responsive data binding framework, which makes it pretty easy to synchronize data with the DOM (Document Object Model). Therefore, Vue.js is particularly suitable for real-time visualization of text annotations. The server side utilizes the Python-based open-source Django framework to build RESTful web services. MySQL database is adopted to organize, store and manage data. The code is available at the GitHub repository https://github.com/FXLP/MarkTool, which also contains a live demo website.
<table>
<thead>
<tr>
<th>Group</th>
<th>Tool</th>
<th>Annotation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>Group-1</td>
<td>WebAnno</td>
<td>1703</td>
</tr>
<tr>
<td></td>
<td>Brat</td>
<td>1870</td>
</tr>
<tr>
<td></td>
<td>DoTAT</td>
<td><strong>1340</strong></td>
</tr>
<tr>
<td>Group-2</td>
<td>WebAnno</td>
<td>1518</td>
</tr>
<tr>
<td></td>
<td>Brat</td>
<td>1767</td>
</tr>
<tr>
<td></td>
<td>DoTAT</td>
<td><strong>1210</strong></td>
</tr>
<tr>
<td>Group-3</td>
<td>WebAnno</td>
<td>1321</td>
</tr>
<tr>
<td></td>
<td>Brat</td>
<td>1503</td>
</tr>
<tr>
<td></td>
<td>DoTAT</td>
<td><strong>1156</strong></td>
</tr>
</tbody>
</table>

Table 1: Annotation time comparison of annotation tools in ACE2005 Dataset. The average annotation time of annotation tool is arithmetic mean value of Time_{avg} in three group. The average annotation time of Webanno is 380s. The average annotation time of Brat is 351s. The average annotation time of DoTAT is 282s.

5 Experiments

5.1 Annotation time

We compare DoTAT with the other two text annotation tools (Brat and WebAnno) on the event annotation task. We randomly select 20 news texts from the ACE2005 dataset (Consortium, 2005), and each text contains at least four sentences. Six students randomly divided into three groups are invited to annotate those texts. For each user, if a tool is used first, more time may be spent since the user is not familiar with the text. To eliminate the influences, each student is given extra time to view the text before the annotation, and each is assigned a different tool using sequences. We separately record the time (in seconds) spent by each group using the three tools when completing 20%, 40%, 60%, 80%, and 100% of the texts. As we can calculate from Table 1, the average annotation time of DoTAT is reduced by 19.7% compared with Brat and 25.8% compared with WebAnno. DoTAT spends less time, since it is time consuming for Brat and Webanno to connect arcs between the trigger and multiple arguments. The mouse movements in the process may be forward and backward. However, DoTAT only needs to select the arguments from a pop up menu on a text span, and the mouse move is typically from left to right.

5.2 Accuracy

We also evaluate the accuracy by comparing with the golden results from ACE20005 data set. The accuracy is computed as:

\[
    acc = \frac{\sum_{i=1}^{n}(Trig_{i,correct} + \sum_{j=1}^{m_i} Arg_{i,j,correct})}{\sum_{i=1}^{n}(1 + m_i)}
\]

where \( n \) is the total number of golden events, and \( m_i \) is total number of arguments in event \( i \). In event \( i \), \( Trig_{i,correct} = 1 \) when trigger is correct, and if argument \( j \) is correct then \( Arg_{i,j,correct} = 1 \). Since annotation quality is too low in real projects with new annotation specifications or new annotators, we often add a particular training process in real application scenarios. Therefore in this paper we design two rounds of experiments, first round is for training and the second round is formal annotation. After round-1, we have a meeting to discuss with annotators about the error-prone events and entities. In Round-2, we select five other most error-prone texts from ACE 2005. As we can see from Table 2, the average accuracy of unreviewed annotations is less than 60% in experiment Round-1. The main reason is that annotators often missed a whole event or missed particular arguments. For example, when using Brat, the proportion of missing events is 33.67% and the proportion of missing arguments is 14.38%. The accuracy of DoTAT is better since it is less possible for DoTAT to miss arguments. When a text span is picked, DoTAT will show all arguments, the pop menu reminds the annotator about the arguments. DoTAT also performs better than Brat and Webanno in Round-2. Besides, the overall accuracy increase in Round-2, which shows that the training process has effects.

In experiment Round-1, the average accuracy of DoTAT’s reviewed annotations reaches 76.2%, which is an increase of 20.9% compared to the average accuracy of unreviewed annotations.

5
Table 2: Accuracy comparison of annotation tools in ACE2005 Dataset. DoTAT-U denotes the unreviewed annotation content of DoTAT. DoTAT-R denotes the reviewed annotation content of DoTAT.

<table>
<thead>
<tr>
<th>Round</th>
<th>Tool</th>
<th>Group-1</th>
<th>Group-2</th>
<th>Group-3</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round-1</td>
<td>WebAnno</td>
<td>44.5%</td>
<td>49.0%</td>
<td>51.7%</td>
<td>48.4%</td>
</tr>
<tr>
<td></td>
<td>Brat</td>
<td>34.5%</td>
<td>44.9%</td>
<td>47.8%</td>
<td>42.4%</td>
</tr>
<tr>
<td></td>
<td>DoTAT-U</td>
<td>45.4%</td>
<td>55.7%</td>
<td>64.8%</td>
<td>55.3%</td>
</tr>
<tr>
<td></td>
<td>DoTAT-R</td>
<td>67.7%</td>
<td>72.6%</td>
<td>88.3%</td>
<td>76.2%</td>
</tr>
<tr>
<td>Round-2</td>
<td>WebAnno</td>
<td>75.48%</td>
<td>82.58%</td>
<td>86.45%</td>
<td>81.5%</td>
</tr>
<tr>
<td></td>
<td>Brat</td>
<td>79.19%</td>
<td>83.87%</td>
<td>85.16%</td>
<td>82.74%</td>
</tr>
<tr>
<td></td>
<td>DoTAT-U</td>
<td>78.71%</td>
<td>86.45%</td>
<td>87.1%</td>
<td>84.09%</td>
</tr>
<tr>
<td></td>
<td>DoTAT-R</td>
<td>93.54%</td>
<td>92.9%</td>
<td>94.84%</td>
<td>93.76%</td>
</tr>
</tbody>
</table>

Table 3: Application of DoTAT.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Task</th>
<th>Annotated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public security</td>
<td>10 types</td>
<td>6 types</td>
</tr>
<tr>
<td></td>
<td>10,000 texts</td>
<td>6,000 texts</td>
</tr>
<tr>
<td></td>
<td>20,000 events</td>
<td>80,000 entities</td>
</tr>
<tr>
<td>Medical</td>
<td>4 types</td>
<td>6,000 events</td>
</tr>
<tr>
<td></td>
<td>300 long texts</td>
<td>18,000 entities</td>
</tr>
</tbody>
</table>

Figure 4: The fraud case annotation example.

age accuracy of DoTAT’s unreviewed annotations. In experiment Round-2, the average accuracy of DoTAT’s reviewed annotations has also increased by 9.67%. It indicates that the review procedure can effectively improve the accuracy. The review procedure not only can complement the missing events and entities, but also reduce the erroneous annotations often caused by the ambiguity of the annotators’ understanding of annotation specifications.

6 Case Study

DoTAT has been used in the annotation projects of three different domains. The details in the public security and medical domains are shown in Table 3. For the criminal case type “fraud” which contains 5 event types and altogether 23 arguments in public security domain, the training process before formal annotation involves four original files and eight annotators. Each file contains 20 texts, which are assigned to two annotators. Consistency checking is performed to inspect the specification understanding of each annotator, and part of the results are shown in Figure 4. We found that the argument “fraud method” scored less than 50% in the four files, because the text span of this argument is not fixed. In the example of Figure 4, some annotator annotated “claim settlement(理赔)” and some annotated “on the ground of claim settlement(以理赔为由)”. Besides, we also found that some simple arguments (such as “name” and “telephone number”) did not reach consistency score of 100%. There are two reasons for this, one is binding an argument to wrong event, e.g. take the “name” of victims as suspects, the other is missing annotation, e.g. “name” of victims appears more than once, but only one place is annotated. Therefore further training is required to solve the disagreement between annotators.

7 Conclusions

The demands for annotation corpus in different domains are rapidly increasing with the development of deep learning. We propose a web-based text annotation tool, DoTAT, which is suitable for domain-oriented complex event annotation. We demonstrate the powerfulness of our tool with experiments and real-world scenarios. We also find training and reviewing are valuable steps to improve the quality of corpus. In the future, we plan to integrate the active learning algorithm into DoTAT to reduce the manual annotation work.
References


