Proceedings of the 13th onference on Language Resources and Evaluation (LRE 2022), pages 4497–4503
Marseille, 20-25 June 2022

QA4IE: A Quality Assurance Tool for Information Extraction

Rafael Jimenez Silva[1], Kaushik Gedela[1], Alex Marr[1], Bart Desmet[1], Carolyn Rosé[1,2], Chunxiao Zhou[1]
[1]National Institutes of Health Clinical Center, Bethesda, Maryland, USA
[2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, USA
{rafael.jimenezsilva, kaushik.gedela, alex.marr, bart.desmet, carolyn.rose, chunxiao.zhou}@nih.gov

Abstract

Quality assurance (QA) is an essential though underdeveloped part of the data annotation process. Although QA is supported to some extent in existing annotation tools, comprehensive support for QA is not standardly provided. In this paper we contribute QA4IE, a comprehensive QA tool for information extraction, which can (1) detect potential problems in text annotations in a timely manner, (2) accurately assess the quality of annotations, (3) visually display and summarize annotation discrepancies among annotation team members, (4) provide a comprehensive statistics report, and (5) support viewing of annotated documents interactively. This paper offers a competitive analysis comparing QA4IE and other popular annotation tools and demonstrates its features, usage, and effectiveness through a case study. The Python code, documentation, and demonstration video are available publicly at https://github.com/CC-RMD-EpiBio/QA4IE.

Keywords: quality assurance, evaluation, discrepancy analysis

1. Introduction

For the past 30 years, benchmark tasks have driven forward progress in the field of Natural Language Processing (NLP). As larger datasets have become the norm in the field, the number of available annotation tools has increased accordingly (Atdağ and Labatut, 2013; Beasley and Manda, 2018; Neves and Ševa, 2021). Though some benchmark tasks leverage training data with naturally occurring labels, common gold standard corpus construction requires substantial annotation effort. Despite advances in pipeline and model development, errors remain likely to occur. Therefore, there is a great need to develop tools that can help correct and improve annotated language resources. A comprehensive quality assurance tool helps guarantee the high quality of language resources and their downstream reliability for evaluation.

In this paper we present QA4IE, a novel tool that addresses an important gap existing in other current annotation tools, namely comprehensive support for quality assurance. Due to the importance of QA in data annotation, most of the existing annotation tools include facilities for error checking (Cejuela et al., 2014; Cunningham et al., 2013; de Castilho et al., 2014; Grosman et al., 2020; Grundke et al., 2016; Klie et al., 2018; Kummerfeld, 2019; LightTag, 2021; Ogren, 2006; Stenetorp et al., 2012), though possibly perfunctory in nature. In contrast, QA4IE offers the following both for annotation and for information extraction: First, it offers automatic detection of potential problems in the annotation results. The user provides formatting standards that are used by QA4IE to validate the annotations and alert the user(s) to deviations from the standard. Second, QA4IE provides measures of agreement between annotators in terms of both token-level and entity-level IRR measurements. Third, based on its comparison between annotations across annotators, QA4IE provides a visual display of the discrepancies between annotators. Fourth, QA4IE provides comprehensive summary statistics computed over annotations, including counts for each type, length distribution, and more. Fifth, users are provided with a corpus viewer that enables viewing annotations of the selected corpus interactively. In addition, a comprehensive QA report, including details of errors and warnings found, text and annotation statistics, annotation performance evaluation, and discrepancy analysis, can be exported in CSV or TXT format.

In the remainder of the paper we will expand upon the desiderata that motivated work on QA4IE. In the next section, we describe in detail the technical foundation for the development of QA4IE. After that we offer a case study illustrating the use of QA4IE. Finally, we provide a competitive analysis, comparing QA4IE to existing tools. The article concludes with a summary and a plan for future work.

2. Annotation Process Support Desiderata

Text annotation specifically refers to the marking of text to highlight structure and substance, for a variety of language tasks that may be operationalized either as classification tasks or sequence labeling tasks. Common annotation tasks include rhetorical structure, concepts and relationships between them, syntactic dependencies, and so on. Annotation tools support the annotation process, enabling far greater efficiency and accuracy than would be possible without them (Ide and Pustejovsky, 2017; Pustejovsky and Stubbs, 2012). In this section we explore the challenges in annotation efforts that render spot checking insufficient and therefore motivate the need for tools with specific affordances for cataloguing potential errors and making them easy to find.

The multi-level structure of natural language provides the opportunity for a variety of error types to occur. In the example of nested named entity recognition (NER) (Nadeau and Sekine, 2007), inconsistencies between annotators may occur in terms of annotation set names and the corresponding annotation schema, sub-entity annotations outside the parent entity annotations, impermissible entity overlap, inconsistent attributes, and so on.

What exacerbates the already difficult QA process is the demand for large scale annotated data, which requires that the annotation work be done by a team of annotators, not just a pair of annotators. In order to ensure the consistency of the annotation work, an inter-rater reliability (IRR) test is performed before the annotators are able to work independently (Artstein and Poesio, 2008). When the IRR is higher than the preset threshold and the main problems that cause inconsistency are resolved, the quality of each annotator's individual annotation work can be assured. Therefore, QA requires the provision of IRR metrics and calculations for teams of annotators.

Due to the constraints of project timeline and budget, it is not always possible to resolve every potential error. Instead, annotation teams must work together to prioritize the most critical issues to address. This requires a QA tool to summarize and categorize annotation inconsistencies and identify the top issues for the annotation team to prioritize. In order to accomplish this, issues must be considered in the context of the big picture of the corpus as a whole. Thus, the QA tool needs to provide summary statistics to support this reflection.

Validation of annotations against a pre-set standard is an incredibly important step that is largely not supported by existing tools. For example, it is typically not trivial to determine whether the original text has been modified in the annotation process, whether a sub-entity is beyond the scope of the parent entity, whether there are overlaps among entities/sub-entities that are not allowed, and so on.

Regarding IRR, although most tools provide the function of calculating IRR, they often provide either the entity-level or the token-level metrics, but not both, even though these two different levels of IRR metrics often provide complementary information and thus need to be considered together. In terms of discrepancy analysis, the presentation of annotation differences is often based on the assumption of one-to-one entity correspondence, which does not always hold (Cunningham et al., 2013). In addition, there is an absence of categorization for different types of discrepancies.

In short, there is a great need for QA4IE, with its comprehensive and powerful QA features for supporting both the quality and efficiency of the annotation process, increasing our understanding of data and annotation, and thus providing an important guarantee for high-quality modeling and evaluation. A more detailed comparison of QA4IE with existing tools is provided in the Competitive Analysis in Section 5.

3. QA4IE System Description

QA4IE addresses all the desiderata discussed in the previous section. Information extraction includes Named Entity Recognition, Coreference Resolution, Named Entity Linking, Relation Extraction, Event Extraction, Terminology Extraction, and so on. In this paper, we use nested NER as an example to illustrate the features of QA4IE. Although the current version of QA4IE focuses on nested NER, these features can be easily applied to other information extraction tasks. QA4IE consists of five main components: Error checking and Validation, IRR, Discrepancy Analysis, Visualization, and Prioritization. In the following, we introduce each component one by one.

3.1 Error Checking and Validation

Error checking and Validation covers both the document and annotation levels. *Document-level validation* is the first and simplest part of QA, which sets the foundation for the subsequent portions. One aspect is to check whether all annotators are using the same basic data format for annotation consistently over the whole corpus. Although text may be pre-processed and distributed to each annotator in the same way before annotation, some annotation tools allow annotators to purposefully or accidentally change text during annotation. Differences in text may also occur if progress is stored locally rather than on a central server. These changes result in false positives in the discrepancy analysis. To address this, QA4IE checks if the text of the same document differs across the versions used by each of multiple annotators. In addition, QA4IE also check consistency in annotation set name across annotators working on the same annotation task.

Annotation level Validation is where annotations are checked for compliance with corresponding annotation guidelines. In order to provide customized services for different annotation tasks, QA4IE offers a structured configuration file, in which the user can provide the following information: (1) the location of the annotation files; (2) the location of the output files; (3) the type of annotation task: QA4IE currently only has the option of sequence labelling (other options, such as text classification, are under development) (4) and the character encoding type. (5) Finally, for sequence labeling, the user can specify the hierarchical structure among entities/sub-entities and which entities/sub-entities are allowed to overlap with each other. QA4IE also checks annotations under default assumptions. For example, by default, the annotations may not be allowed to go beyond the range of the text; or, the end index of an entity/sub-entity may not be allowed to be equal to or less than its starting index. Annotation offsets that occur outside of the scope of the text may seem impossible. This is true for most original annotations. But annotation results often need some post-processing, such as converting format, filtering irrelevant content, etc. If the post-processing process allows modifying the indices of annotations, this unexpected error may arise.

3.2 IRR

IRR, which is also called inter-annotator agreement (IAA), is commonly used to measure the quality of annotation work. Assuming that the annotations from each annotator have been validated, the next step is to ensure some level of consensus among annotators. Cohen's kappa coefficient (Artstein and Poesio, 2008) is commonly used to measure IRR for classification tasks because it is a chance-corrected metric. For sequence labeling, where chance agreement calculation is still an open problem, F1 score is usually used to approximate the IRR (Cunningham et al., 2013).

3.3 Discrepancy Analysis

IRR is just a measure of agreement that defines quality in terms of reproducibility across annotators. As a next step, the *Discrepancy Analysis* functionality is used to help the annotation team find out where the problems are, the reasons for the problems, and to distinguish between more and less important problems as part of their prioritization process. The discrepancy analysis in existing annotation

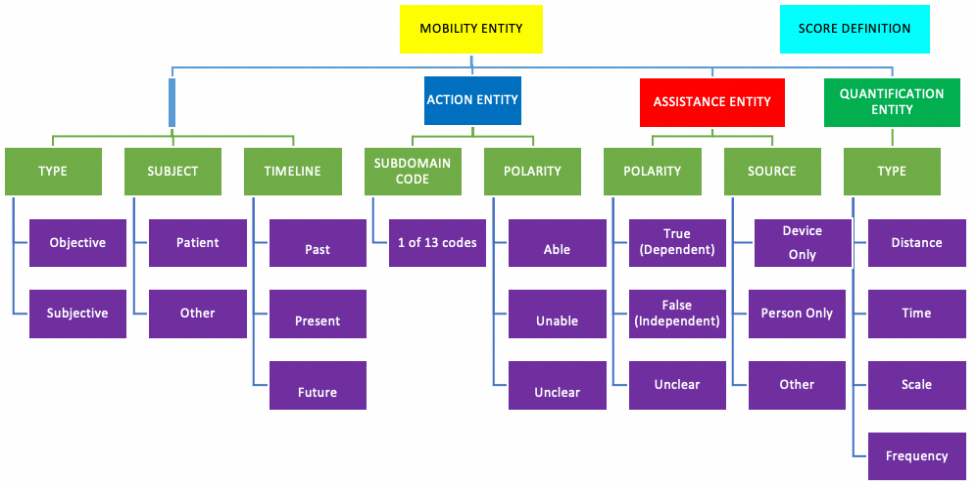

Figure 1: Mobility information structure.

software is typically only a display of differences based on the assumption of one-to-one correspondence, i.e., each unit of one annotator's annotation can be matched to a unit of another annotator's annotation if both of them annotate the corresponding unit in the same text (Cunningham et al., 2013). However, this assumption does not always hold. We find that for the annotation of long text span entities, the annotation style of each annotator is different, which often results in a many-to-many correspondence situation. This is because some annotators are accustomed to labeling a continuous long text span as several short entities, while others may be used to combining several adjacent short entities into one long entity.

3.4 Visualization

Visualization is then offered beyond simply pinpointing specific discrepancies between pairs of annotators. The display for checking one-to-one correspondences does not fully demonstrate the similarities and differences across multiple annotators. By partitioning the annotations of multiple annotators into clusters based on annotation overlap, QA4IE compares the annotations of multiple annotators within each resulting text span cluster, so that these more complex sets of differences between annotators can be seen at glance. To provide better feedback to the annotation team, QA4IE supports assignment of these clusters into the following categories: (1) not every annotator has annotated this; (2) split/combine, i.e., an entity annotated by one annotator overlaps with multiple entities of another annotator; (3) partial match; (4) corresponding entities have the same text span, but their attributes are different. QA4IE compares the discrepancies of each pair of annotators and the whole annotation team, and sorts the various types of discrepancies for the annotation team's reference.

3.5 Prioritization

Prioritization can be supported by enabling an annotation team to view identified issues in the context of the composition of the corpus. QA4IE also tallies the annotations of each annotator to better understand the characteristics of the task and the annotation style of each annotator. Summary statistics include: (1) the number of various types of entities in each document; (2) number of entities with specific attributes; (3) length of entities, and (4) the length of entities with specific attributes.

We will illustrate all features with details in the case study below.

4. Case Study

The current QA4IE is a Python based command-line tool that is intuitive and accessible. Anyone familiar with the command line interface can quickly master all of QA4IE's features. The user needs to provide the input folder for annotated GATE XML files, the output folder for QA reports, and the annotation schema specifications in a configuration file. GATE supports a range of file formats by default, including XML, and many more through plugins[1]. Since the default format for QA4IE is GATE XML, we suggest using GATE as a preprocessing step to convert your annotations if they're in a different format. Here we use some synthetic clinical notes as an example to demonstrate the application of QA4IE to a sequence labeling task. This task is to extract mobility information from the clinical notes by three annotators. The mobility information is organized as shown in Figure 1.

Note mobility related annotations have a hierarchical structure, and that the schema file lists only the associated entities, sub-entities, and corresponding attributes, and does not specify the nested structure between entities and sub-entities. Therefore, we list all the entities and sub-entities in the configuration file with the identifier []. The dependency of the sub-entities and their parent entities is indicated by "sub_entities=". In Figure 2, the mobility entity contains three types of sub-entities: Action, Assistance, and Quantification. Based on the nested structure given in the configuration file, QA4IE can detect errors in the sub-entity that exceed its parent entity and output them to the quality assurance report. The types of entities/sub-entities that are allowed to overlap with each entity/sub-entity are listed after "overlaps=", and the types of entities/sub-entities that are not listed are not allowed to

[1] https://gate.ac.uk/sale/tao/splitch5.html#x8-920005.5

overlap with the entity/sub-entity by default. In addition, we also specify the type of task in the configuration file, i.e., sequence labeling or text classification, and character encoding type.

Once the configuration file is ready, we can directly run QA4IE to perform the QA task. The main menu is shown in Figure 3. Regardless of the option selected, users have the flexibility to select specific text, annotators, entity/sub-entity, and features according to their needs.

```
[required]
annotations_dir = /Documents/Quality Assurance Tool/QA4IE-main/data/annotations
output_dir = /Documents/Quality Assurance Tool/QA4IE-main/output

task = sequence_labelling
encoding = UTF-8

[Mobility]
overlaps = Score Definition|Instructions/Questions
sub_entities = Action|Assistance|Quantification
features = Timeline:=:Past|Present|Future||Type:=:Objective|Subjective||Subject:=:Patient|Other

[Assistance]
overlaps = Assistance|Action|Quantification
features= Polarity:=:True|False|Unclear||Source:=:Person Only|Other|Device Only

[Action]
overlaps = Assistance|Action|Quantification
features = Subdomain Code:=:d410|d415|d420|d430|d435|d440|d445|d450|d455|d460|d470|d475|d480||Polarity:=:Ab

[Quantification]
overlaps = Assistance|Action|Quantification
features=Type:=:Distance|Time|Scale|Frequency

[Instructions/Questions]
overlaps = Score Definition|Instructions/Questions|Mobility

[Score Definition]
overlaps = Score Definition|Instructions/Questions|Mobility
```

Figure 2: Configuration file for mobility information extraction.

```
QA4IE Main Menu

Please select an option:

Document Validations
Annotation Validations
Statistics
Evaluation
Discrepancy Analysis
Generate Reports
Corpus Viewer
Annotation Schema
Help

Exit
```

Figure 3: QA4IE main menu.

Document Validation includes two functions: “Text Differences” and “Set Name Difference”. If errors have been found, users can generate reports to output the text and indices of text fragments where discrepancies have been found, as well as differences in set names.

```
Document Validations
File : corpus Annotator Pair : team
Please select an option:

Text Differences
Set Name Differences
Generate Report


 Text differences found
```

Figure 4: Example of Document Validations

Annotation Validations have several sub-options. "Annotation Overlaps" detects overlapping entities/sub-entities that violate regulation. "Subentity Boundaries" finds sub-entities that are beyond the scope of their parent entities, while "Annotation Boundaries" checks if any annotation falls outside the annotation set. "Zero Length Annotations" and "Negative Length Annotations" catch entities/sub-entities whose end indices are less than or equal to the starting indices. "Document Scope" identifies annotations that are beyond the scope of the full text. "Validate Schema" checks if there are annotations that are not defined in the schema.

```
Please select an option:

Annotation Overlaps
Subentity Boundaries
Annotation Boundaries
Zero Length Annotations
Negative Length Annotations
Document Scope
Validate Schema
Generate Report


 Annotation overlaps found
```

Figure 5: Example of Annotation Validations

Statistics is used to collect information on various distributions, ~~i~~including the number of various types of entity/sub-entity and their lengths in terms of mean, variance, and extreme values. The choice of types can be either entity/sub-entity without regard to feature, or entity/sub-entity with a specific feature. Basic statistical information is displayed directly on the screen. A more complete version can be exported to the output folder

```
Statistics
File : corpus Set Name : all_sets Annotator : team Entity : all_types
Please select an option:

Entity Distribution
Entity/Features Distribution
Entity Token Length Distribution
Entity/Features Token Length Distribution
Generate Report


 Action : 72
 Assistance : 10
 Instructions/Questions : 1
 Mobility : 76
 Quantification : 48
 Score Definition : 12
```

Figure 6: Example of Statistics

Evaluation provides both token-level and entity-level metrics because they provide complementary information.

```
Token Level
File : corpus Set Name : all_sets Key : anno1 Response : anno2 Entity : all_types
Please select an option:

Evaluate
Generate Report


------------------------------------
          true_response false_response
true_key    195            100
false_key   111            6608
------------------------------------
precision : 0.985
recall : 0.983
f1 : 0.984
```

Figure 7: Example of Evaluation

Figure 7 shows the token-level metric as an example, giving the confusion matrix, precision, recall, and F1 score for a selected type of entity/sub-entity.

Discrepancy Analysis is another highlight of QA4IE. Our Discrepancy Analysis has two unique features. As shown in Figure 8, QA4IE automatically categorizes discrepancies by text span and feature. This allows the annotation team to understand which are the main discrepancies that need to be resolved first.

```
Discrepancy Analysis
File : corpus Annotator Pair : team
Please select an option:

Compare
Generate Report


 text discrepancies
   not annotated by all : 55
   length : 50
   split combine : 26
 feature discrepancies
   timeline : 8
   polarity : 4
   subdomain code : 4
   type : 4
   subject : 5
   source : 1
```

Figure 8: Discrepancy Analysis