CSAF - a community-sourcing annotation framework

Jin-Dong Kim and **Yue Wang**
Database Center for Life Science (DBCLS),
Research Organization of Information and Systems (ROIS),
2-11-16, Yayoi, Bunkyo-ku, Tokyo, 113-0032, Japan
{jdkim|wang}@dbcls.rois.ac.jp

Abstract

This paper presents a community-sourcing annotation framework, which is designed to implement a "marketplace model" of annotation tasks and annotators, with an emphasis on efficient management of community of potential annotators. As a position paper, it explains the motivation and the design concept of the framework, with a prototype implementation.

1 Introduction

Corpus annotation is regarded indispensable for the development of language-processing software and technology, e.g., natural language processing (NLP) and text mining. Nevertheless, the high cost required for finding and maintaining human annotators often hinders the development of various corpus annotation. For an annotation project, annotators, e.g., domain experts, need to be recruited, trained, then deployed for actual annotation. After the annotation project is over, usually they are dismissed. The same cycle then needs to be repeated for a new annotation project. In this setup, the recruitment and training of annotators actually take non-trivial cost.

Recently, crowdsourcing, e.g., Amazon Mechanical Turk (MTurk, hereafter), is gaining a big attention as a source of finding intelligent human labor. For corpus annotation also, the usability of MTurk has been explored (Callison-Burch and Dredze, 2010; Buzek et al., 2010; Little et al., 2009). There are also other efforts to achieve a large-scale annotation based on community-wide efforts (Ide et al., 2010), which shows current trends toward systematic incorporation of contributions from a community rather than from a small group.

In this work, we propose a community-sourcing annotation framework (CSAF, hereafter) which defines the components and protocol of a computer system to enable community-sourcing annotation. It is similar to MTurk to some extent in its concept, but it is more specifically designed for corpus annotation tasks, particularly for those which require special expertise from annotators, e.g., domain knowledge. With "community", it means a group of people who are regarded as qualified potential annotators for a specific type of annotation tasks. For example, for semantic annotation of biological literature, e.g., PubMed, graduate students of biology may be regarded qualified, and will be expected to form a community of potential annotators. The goal of CSAF is to provide a framework of computer system to enable an effective and efficient maintenance of such communities, so that when an annotation project is launched, available annotators in a community can be immediately found and deployed. It is also expected that the effect of training can be accumulated in the community.

With the background, in this position paper, the the core design concept (section 2) and the specifications and a prototype implementation (section 3) of CSAF is discussed.

2 Community-sourcing annotation framework (CSAF)

CSAF consists of four components: annotation editor (AE), task server (TS), task manager (TM), and community manager (CM). Among them, the first

Proceedings of the 6th Linguistic Annotation Workshop, pages 153–156,
Jeju, Republic of Korea, 12-13 July 2012.

three, which are shown in figure 1, are actually required for any usual annotation project, no matter how explicitly they are implemented. The last one, CM, being integrated with the others, enables community-sourcing annotation.

2.1 Components for usual annotation

An AE provides annotators with a user interface (UI) for creation or revision of annotations. This component is often the most explicitly required software for an annotation project.

A TS takes the role of assigning annotation targets, e.g., documents, to annotators. Often, the assignment is performed manually by the organizers, particularly when the annotation projects are in a small scale. However by automating it, the assignment could be achieved in a more systematic and error-free way. A possible implementation may include a sequential assignment with a periodic overlap of some documents over the annotators for quality control, e.g., inter-annotator agreement rate. A TS may be regarded as manifestation of an assignment policy while an AE as manifestation of an annotation scheme.

A TM is to manage the progress of annotations performed by an individual annotator. Also, the management is often performed manually, but provision of a proper tool should enhance the management substantially. Together with an AE, it provides annotators with an annotation environment. As usually annotators are not experts of computer systems, provision of a convenient annotation environment is closely related to the productivity of annotation practice.

Although the three components do not include any notion of community-sourcing, separation of the three eases incorporation of an additional component, community manager which will be explained in next section.

Figure 1 illustrates how the three components work with together over the standard HTTP protocol in CSAF. An annotator on an annotation task will work with a TM and AE. The annotator may begin the annotation by requesting a document to the TS (1). On request, the identifier of the annotator needs to be notified to the TS, so that the TS can perform an assignment considering the annotators. The annotator then can open the document in the AE (2), and work on annotation. after a session of annotation, the resulting annotation will be downloaded to TM (3). The steps (2) and (3) may be repeated until the annotation is completed. When complete, the final annotation will be uploaded to the TS (4).

2.2 A component for community-sourcing

Figure 2 illustrates how an additional component, CM, enables community-sourcing of annotation. A CM plays like a job market where annotators and annotation tasks are registered, and associations, e.g., recruitment, between them are made. A possible scenario would be as follows: whenever a new task is registered, it is notified to the registered annotators; available annotators will apply to working on the task; and on approval of the task organizer, the association will be made. For each association, a TM is created for the management of the progress of the annotation by the annotator on the task. Once a TM is created, annotation by an individual annotator is carried over in the way described in the previous section

3 Specifications and implementations

In CSAF, all the four components are designed to be web services that will communicate with each other over the standard HTTP protocol.

3.1 Annotation Editor

An AE is supposed to take parameters (by HTTP POST) for two objects, a document and a set of pre-annotations, to enable production of a new set of annotations (by annotators), and to allow download (by HTTP GET) of the newly produced annotations. For the parameters, the document and the pre-annotations themselves may be passed over in an XML or JSON format. Alternatively, the URLs of them may be passed so that they can be read by the AE, when they are accessible from the network. The IO interface is intended to be minimal and flexible so that many existing web-based annotation editors can be integrated in the framework at a minimal cost. As a reference implementation, a simple AE that supports a named entity-style annotation is implemented. Figure 3 shows a screen-shot of it.

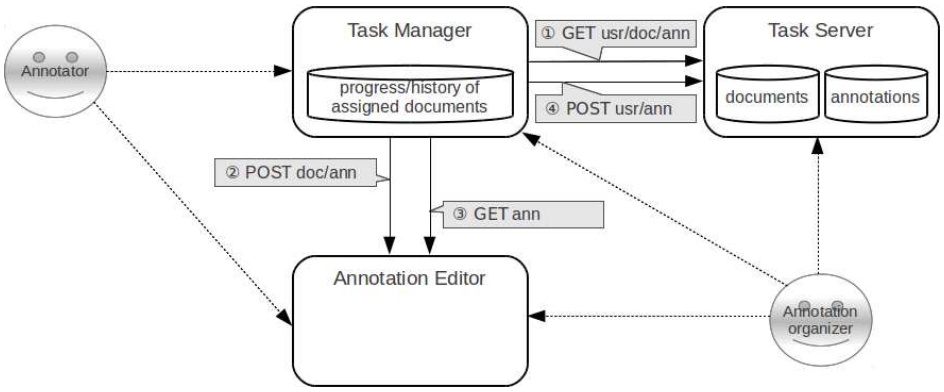

Figure 1: Components for usual annotation tasks

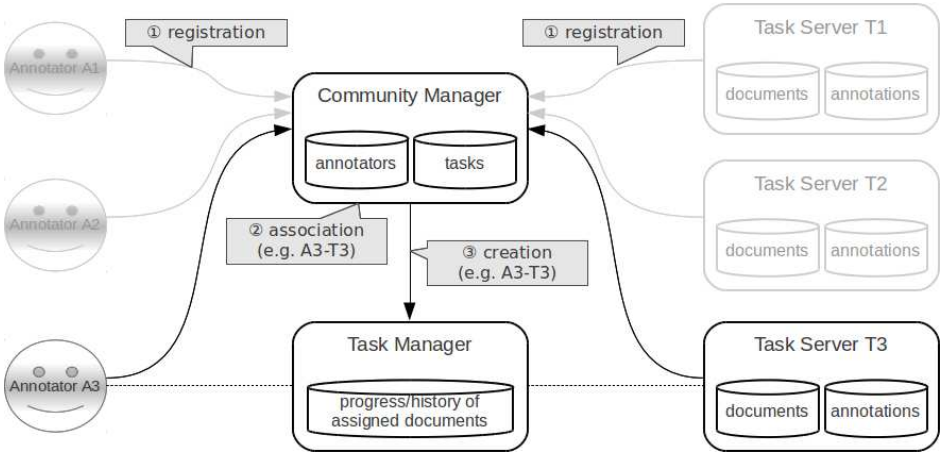

Figure 2: The role of community manager for community sourcing

3.2 Task Server

A TS is supposed to provide (1) annotation guidelines and (2) a document dispatcher, and to take back a new set of annotations (by HTTP POST). Annotators will access the guidelines (by HTTP GET) for reference before application and during annotation. The document dispatcher is an implementation of the organizer's strategy on how to assign documents to the annotators. On request from TM (by HTTP GET), a document is assigned to the annotator, optionally with a set of pre-annotations.

3.3 Task Manager

A TM is created for each association of an annotator and a task, based on the information supplied by the task organizer. It communicates with a TS to get a document to annotate, and with an AE to produce a set of new annotations. It is the responsibility of a TM to maintain the progress of annotation, e.g., the documents that have been or to be annotated.

3.4 Community Manager

As a community manager, account management, e.g., registration or unsubscription, is a fundamental function of CM. The users of a CM are either *annotators* or *task organizers*[1]. The task organizers can register annotation tasks to the CM. Figure 4 shows an example of task registration. Note that URLs given for the *job request* and *editor* specify how the required parameters, `annotator_id`, `document_url`, and `annotation_url` can be passed to the TM and AE.

[1]There is also a superuser who has all the privilege to modify or delete all the other accounts.

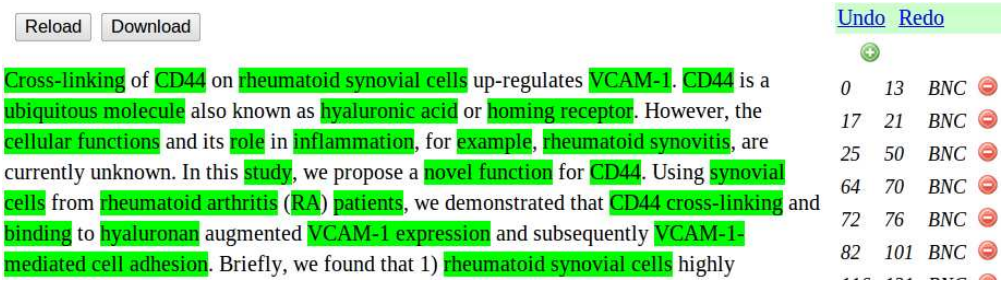

Figure 3: An annotation editor with base-noun-phrase annotations

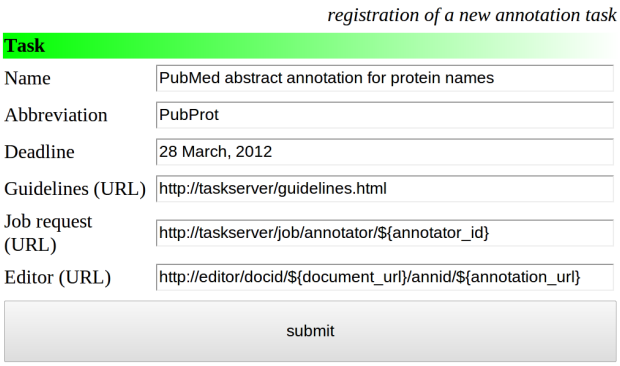

Figure 4: Registration of a new task to the prototype community manager

On registration of a new task, more than one annotators can be associated with the task through a negotiation. For each association of an annotator and a task, an instance of TM is created based on the information shown in Figure 4.

4 Discussions and conclusions

While the importance of corpus annotation is widely accepted, the low productivity of annotation often discourage production of new annotation. In this work, we present a community-sourcing annotation framework (CSAF) with the goal to reduce the cost for recruitment and also training of annotators. A prototype system of CSAF is implemented as a testbed, with a simple annotation editor as a reference implementation. The prototype system will be released to the public.

There is a much room for improvement in the framework and the prototype system. For example, the format of annotation is not yet specified, and it is currently the organizers responsibility to prepare a pair of TS and AE that can work with each other. The way of negotiation for recruitment and the rewarding system are also not yet specified. We plan to keep developing CSAF, and hope this position paper to facilitate discussions and collaborations.

Acknowledgments

This work was supported by the "Integrated Database Project" funded by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

Olivia Buzek, Philip Resnik, and Benjamin B. Bederson. 2010. Error driven paraphrase annotation using mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 217–221, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 1–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nancy Ide, Collin Baker, Christiane Fellbaum, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the ACL 2010 Conference Short Papers*.

Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2009. Turkit: tools for iterative tasks on mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 29–30, New York, NY, USA. ACM.