Annotator's Guide

Note: Annotators should also read the Adjudicator's Guide, so that they understand the larger picture of how the annotation team works and what the adjudicator's role is.

What is Annotation?

Annotation is the process of applying a descriptive or analytic notation to raw language data. It is the process of making normally implicit information explicit. In most of the projects undertaken in our lab, we are doing *syntactic and semantic text annotation*. *Text annotation* means we are working on language packaged in the form of a written text, such as a folktale, religious story, or newspaper article. *Syntactic annotation* means that we are making explicit aspects of the syntax of the language, such as the phrase boundaries, parts of speech, grammatical structure, and so on. *Semantic annotation* means that we are making explicit aspects of the meaning of the language, such as what the words mean, who the characters are, and what events are described.

There are many possible reasons for doing annotation, but they all fall into two basic categories. The first reason is to capture, in an explicit and unequivocal representation, some aspect of a person's understanding of a text, so that this information may then be used to train or test computer systems that aim to emulate people. The second reason is to construct an understanding of some aspect of a text which cannot yet be automatically determined by computer.

The Story Workbench

In this annotation project we will be using a computer program called *The Story Workbench* to perform annotation. The Story Workbench is a tool that facilitates the collection of text annotations. It is similar in appearance to a word processing program: it has an editor area where one can enter and modify text, menus and buttons for performing operations on that text, and a variety of side views showing supplementary information. In great contrast to a word processor, however, it allows you, the annotator, to specify what the text 'means' to you, i.e., to annotate it. This annotation is not usually done from scratch. Rather, the Story Workbench uses off-the-shelf natural language processing (NLP) technology to make a best guess as to the annotations, presenting that guess (if necessary) to you for approval, correction, and elaboration. This is neither fully manual, nor fully automatic, and thus will be called semi-automatic annotation.

To help you prepare for annotation you will be provided not only with this guide, but also a guide for each of the annotation schemes which you will be expected to annotate. Remember the following:

If you ever lose your guides, or need to be reminded of the Story Workbench instructions, go to the **Help** menu and select **About Story Workbench**. In that dialog there is a URL which will direct you to all available guides and instructions. This is shown in Figure 1.

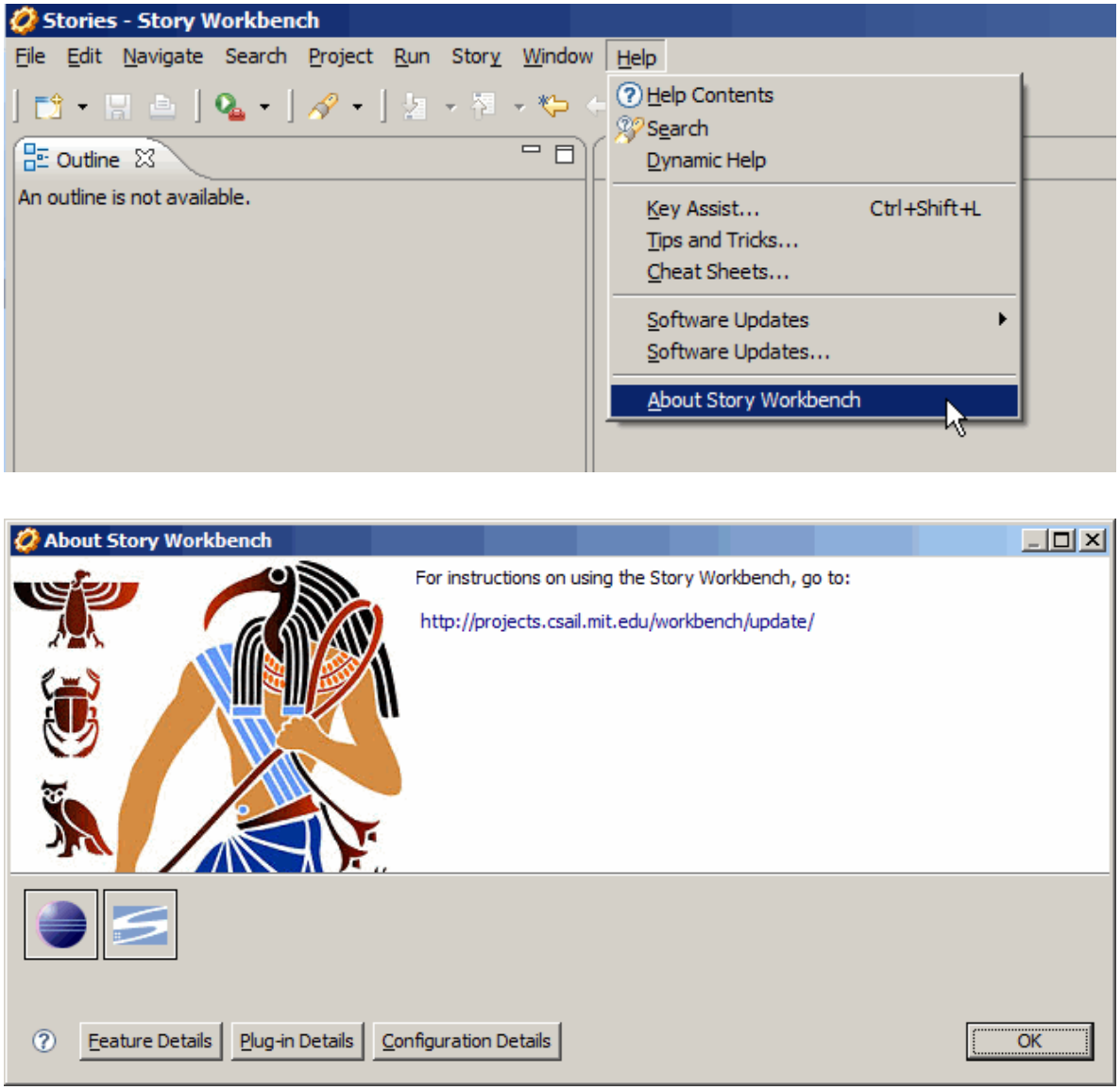

Figure 1: The About dialog.

Workspace

The Story Workbench is very simple to install. First, ensure that you have Java installed on your machine. Second, download the Story Workbench installation zip file archive and uncompress it to your computer's hard drive. The workbench executable is inside the folder. When you first run the workbench, it will prompt to choose a *workspace*. The workspace is a folder on your hard drive that contains all the projects which contain all the files that show up in the Story Workbench's file navigator. If you choose the default workspace location, the workspace folder will be *inside* the Story Workbench installation folder.

When you create new projects inside the Story Workbench navigator view, either from scratch or by checking them out of the repository, each project will correspond to a folder inside of your workspace folder.

Updates

The Story Workbench is designed to accept updates without requiring a full re-installation. You can check for updates manually by choosing the **Software Updates** menu option, or you can set your installation to periodically look for updates. The recommended setting is to set your installation to automatically check for updates whenever the program is started.

Annotation Work Flow

You will not be annotating alone – you will annotate texts in cooperation with two other people, your *annotation team*. An annotation team consists of two annotators (one of whom is you) and an *adjudicator*. The adjudicator is generally a more experienced annotator who is responsible for the overall quality of the annotations being produced by your team. The practice of annotating the same text twice is called *double-annotation*.

Both you and the other annotator on your team will annotate exactly the same texts. For any given week of annotation, your adjudicator will set a target set of texts to annotate, and you will schedule a *merge meeting* for some point later in the week. Using the Story Workbench you will then annotate those texts on your own schedule.

The whole team comes together at the merge meeting. The adjudicator takes your annotations and compares them with those of the other annotator, and the three of you discuss any discrepancies. The adjudicator makes corrections to a master file which then becomes the *gold standard* annotation for that text. This pattern defines the work flow for annotation: a merge meeting, followed by a week of annotating, repeat.

When annotating, you should be very careful not to change any aspects of the text other than the annotation scheme that you are assigned. This means, especially, that you should not edit the text itself in any way.

Never edit the text itself, or any other layers of annotations other than the annotations you are assigned to annotate.

If you find, for example, typographical or formatting errors, notify your adjudicator and supervisor. These errors will be corrected by the supervisor and those changes will be integrated with your texts via a special procedure. If you find errors in previous layers of annotation, also notify your adjudicator and supervisor, and those errors will be corrected when necessary.

Repositories & Version Control

The Story Workbench is equipped with a *version control* system that enables the annotation supervisor to distribute files to you, keeps track of which files you have modified, keeps track of all your changes, and allows you to save all your work to a remote server. Saving your files to a remote server, called the *repository*, ensures that when you make changes to files they can be reversed at a later time, and if your computer or Story Workbench installation suffers a catastrophic failure, only your most recent, unsaved work is lost.

There are three basic operations for version control: *checkout*, *update*, and *commit*.

Checkout

When you are first setting up your Story Workbench installation, you will need to *checkout* a copy of one or more remote repository folders onto your hard drive. This local copy is called your *working copy*. You will have a user folder assigned to you. Within that folder, there will be one or more subfolders. Each of these is an individual project which you must checkout into your Story Workbench workspace using the "SVN Repositories" view. Suppose, for example, your username is "user003" and you have two folders in your repository user directory: *semroles* and *senses*. Each of these is a project that should be checked out into your workspace, as shown in **Error! Reference source not found.**. A file that is a checked-out copy of a file in a repository will have a little repository sub-icon on it, in the shape of a small, orange cylinder.

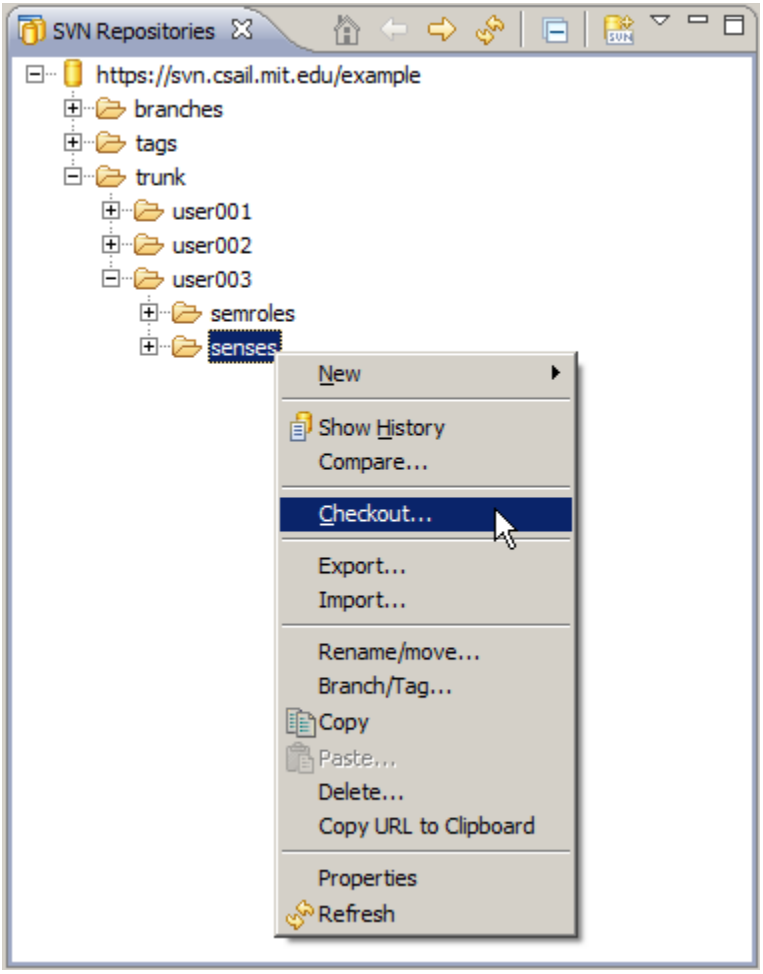

Figure 2: Checkout a project using the context menu.

Commit

Once you have a project checked out into your workspace, you are ready to begin annotation. Do your annotation tasks, following the guidelines laid out by your annotation scheme. This will result in changes to one or more files. If you have changed version controlled files, these files will have a

black dot next to them, indicating they have changed since you last saved your changes to the repository. Black dots on your files or folders mean that your changes have **not** been saved to the repository. To commit those changes to the repository, open the context menu on the project root (not the file itself) and select "Commit…" as shown in Figure 3.

Always commit the **whole** project, and not individual files. If you commit files piecemeal, it makes it more difficult for the supervisor to see what work you are doing, and easier to forget to commit a file.

The commit dialog will prompt you for a message when committing. Make sure to enter an informative message. Good messages are full sentences containing a clear description of what has changed since your last commit. Examples:

- "Finished annotating events on this file."
- "Corrected new errors on story001; finished annotating story002."
- "Fixed previous inconsistencies on part of speech annotation for these committed files."
- "Moved these files from the 'todo' to the 'done' folder."

Bad messages are incomplete sentences that do not tell the reader what is happening. Examples:

- ""
- "…"
- "asdf"
- "done"
- "finished"
- "committing these files"

Commit messages are extremely important for tracking the progress of the work: they are saved permanently with the changes you make. Every time you make a commit, your supervisor gets an email and he reads your message, so make your messages descriptive and informative!

Always write an informative commit message. **Never** leave a commit message blank.

You do not need to include your username in the commit message – this is automatically included by the software. You also do not need to include the names of the files being committed as long as it is unambiguous what things are being done to which files.

If you do not commit your files regularly, the supervisor and adjudicator do not know whether you are doing your work and cannot access your results. Your work is also not insured in the case of a catastrophic computer failure. Commit your work often to the repository. At a minimum, this means at the end of every day or working session, whichever comes first. Whenever you finish with a single file, commit those changes to the repository. Never leave uncommitted files overnight.

Commit your changes to the repository at least once a day, if not more often.

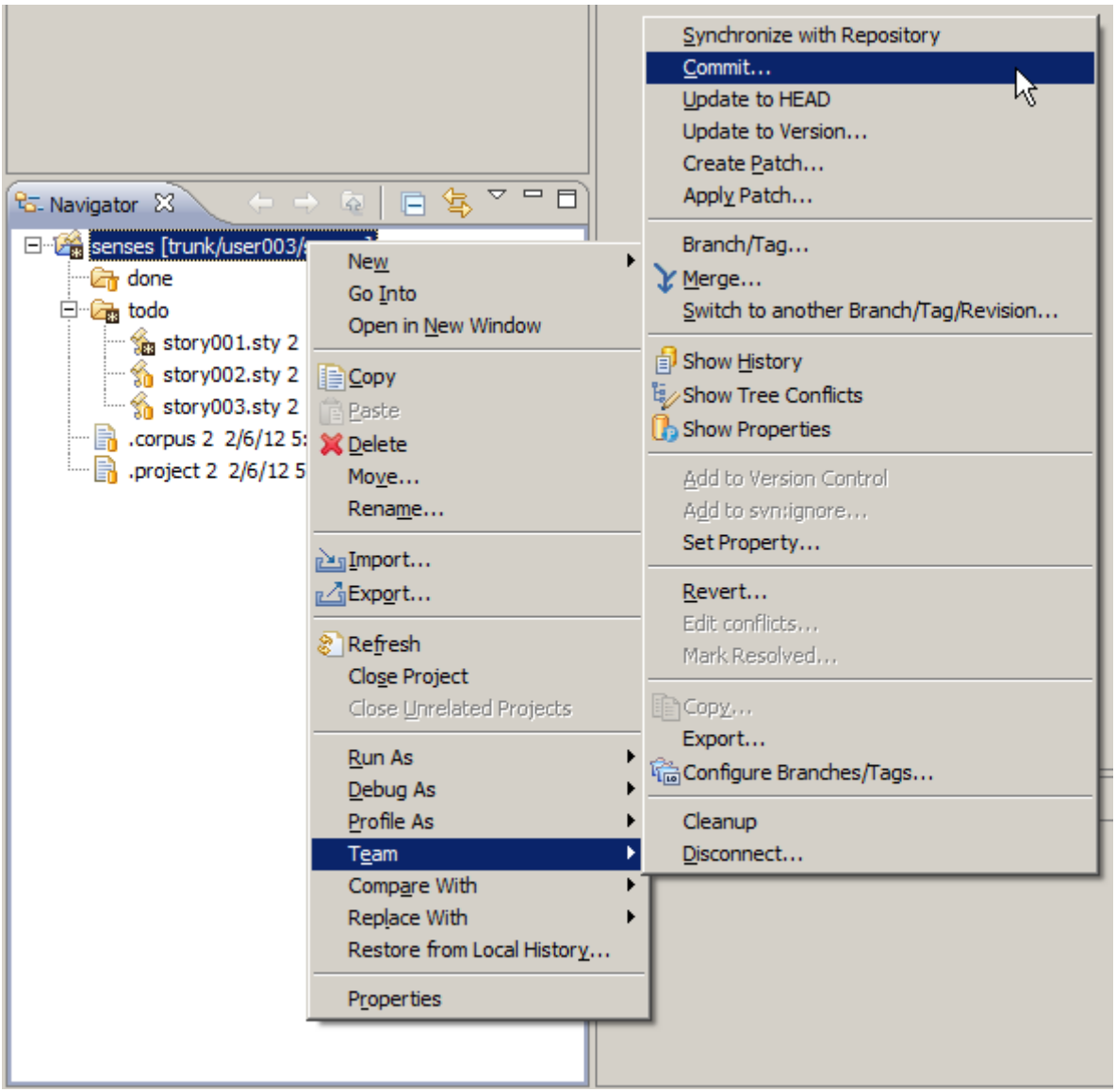

Figure 3: Committing changed files using the context menu. Note the small orange cylinder on each file and directory noting it is a version-controlled object, and the black dot indicating a changed file. To the right of each file is a revision number indicating the version of the file.

Update

Sometimes, someone else makes a change to your files or projects and commits those changes to the repository. This may happen when your supervisor adds new files for you to annotate, or corrects an error in files already in your projects. To obtain these changes, you must *update* your project by selecting the "Update to HEAD" item. This downloads the changes to your local working copy. If you try to commit changes to a file for which you do not have the latest version, the versioning system will return an error, and tell you that your working copy is out of sync with the repository. If this happens, consult your supervisor for assistance. It is good practice to update your project when you begin working.

Practice Good Repository Hygiene!

Good repository hygiene means that your files are never in conflict with the repository for any extended period of time. You maintain good repository hygiene by updating your project before you begin working for the session, and committing your changes, with an informative commit message, when you are done with your work session.

Update on rise, Commit before bed, makes an annotator healthy, wealthy, and employed.

Performance Metrics

How do you know if you are doing your job well? That is, how do you know if your annotations are actually correct? There are a number of immediate measures of annotation quality that are specific to every particular annotation scheme. You should refer to your annotation guide for more information. In general, however, the primary way we measure quality of annotations is by calculating one or more *inter-annotator agreement measures* between your annotations and the other annotator on your annotation team.

At the beginning of each merge meeting your adjudicator will measure the inter-annotator agreement between your texts and your annotation partner. Your goal is to get these numbers as high as possible, without directly colluding with your annotation partner to produce the same annotations.

Annotator Responsibilities

The following section lays out annotator responsibilities. Please read this section carefully as it contains information on not only what is expected of you, but also how to avoid administrative headaches and get paid in a timely manner.

Quality of Annotation

The primary responsibility of an annotator is to produce consistently high quality annotations at a reasonable rate. You should strive for high inter-annotator agreements with your annotation partner, and participate thoughtfully in the discussions of annotation differences at the merge meetings.

The primary responsibility of all annotators is to produce high quality annotations. Everything else is secondary.

Feedback on Annotation Guides and Software

The second most important responsibility of an annotator is to provide feedback on the annotation schemes, guides, and software. If you see something, say something! If you notice a common sort of error occurring in your annotations, mention this to your adjudicator or supervisor; there may be a way of automatically detecting or correcting those types of errors. If you find parts of your annotation guide confusing or unintuitive, or find typographical errors, notify your supervisor. If

you find a bug in the software, report it: it will be fixed as soon as possible not only for you, but for all the other annotators, and everyone will be grateful.

Work Schedule

Annotation is a flexible job, but does require careful time management. You should maintain a consistent pace of work. Never, ever cram all your annotation for a week into one session. It is better to ask for your merge meeting to be rescheduled and come to the meeting having worked consistently, rather than to come with poor annotations.

Meetings are scheduled by each annotation group individually. The supervisor is aware of the schedule and may drop in to see how things are going, but it is the responsibility of each team to keep themselves on track and do their assigned work. Annotators should arrive on time to the meetings prepared. Get the phone numbers of all the members of your annotation so you can notify them if you will be unexpectedly late, or must cancel at the last minute due to an unforeseen circumstance. Meeting times are not assigned – they are scheduled at the convenience of all members of the team. Therefore consistent tardiness is not acceptable.

Make sure you are well rested before you begin annotation and before each merge meeting. While annotation is not a difficult job, it does require careful concentration. The more awake and fresh you are, the better your annotations will be and the higher your inter-annotator agreements.

Respect and Attention

Most of your time during annotation is spent by yourself on your own schedule. However, you are responsible for being an interactive member of the team at merge meetings. Every team has only three members, so this means **you are a critical to your team's success**! This means that if you don't understand something, or disagree with the interpretation taken by the adjudicator or other annotator, speak up. The more you voice your opinion and ask questions, the better you will understand your work the higher quality your annotations will be.

There is a note facility integrated in the Story Workbench, where you can add arbitrary notes to any annotation. Use this facility freely to explain any unusual or controversial annotations you have made, or to write down questions to be address during merge meetings.

When discussing annotations with your team, always be respectful. Remember, the goal of the project for any one annotator to be right, but for the annotations to be right. Annotations are not personal. If, in the case of a discrepancy, the adjudicator chooses the other annotator's answer, don't be offended: it is because the adjudicator thinks that is the answer that produces the annotation most consistent with the meaning of the text and the annotation scheme, **not** that they dislike you. On the other hand, the adjudicator should be able to justify their answer to your satisfaction.

There should be some amount of discussion for most discrepancies. However, it should not go on forever: the annotation team has a goal to get through a certain number of words each week. Therefore the adjudicator will, at some point, end the discussion and make a decision, possibly with an appended note for the supervisor to review. If the discussion reveals an especially difficult

decision, this is an indication that the annotation scheme might need to be adjusted, or a new rule added.

Keep your Supervisor and Adjudicator Informed

Annotators should make sure to keep their adjudicator and supervisor informed when they are sick and won't be able to make a meeting or do their work that week. Any major changes of schedule or availability should be discussed with supervisor if they negatively impact your ability to do your annotation. You should also notify the supervisor and your team members if your email or phone number changes. Changes of address should be reported to the MITemp administration.

Reporting your Hours

Each employee is responsible for keeping track of the number of hours they work. Work time includes time spent annotating at home, as well as annotation merge meetings, but does **not** include time it takes to travel to the merge meeting.

Keep track of your hours to the nearest quarter-hour. You should expect your work hours to vary somewhat from week to week. However, there is a general range of time that the supervisor is expecting to see you work – if your reported hours diverges significantly from the expected time (higher **or** lower) be prepared to explain why this is so.

Because of problems we have had in the past with unreported time, hours worked **must** be reported within 7 days of the end of a work week. Work weeks end on Sundays. This means that you must have submitted your time sheet for a week by the end of the next Sunday. Hours not reported within 7 days will not be paid for except by explicit permission of the supervisor. Preferably you will submit your time at the end of the Sunday which ends that work week – that is, you will not wait a week, but submit your hours immediately.

Time sheets must be submitted within 7 days of the end of a work week.
!!! Time not submitted within 7 days will not be paid !!!

We have also problems in the past with MITemps not paying annotators the correct amount. Be aware of your hourly rate, how much you have worked, and how much you should be getting paid. If your paycheck does not match that expectation, bring up the issue with your supervisor.

Problems

If you have a problem with your work, or with another member of your team, bring up the issue with your supervisor. Happy annotators make for quality annotations, so don't hesitate to bring up issues with your supervisor so they can be resolved as soon as possible.