SER Practicals - Day 1

Git and the command line can seem scary at first but the best way to dispel that fear is by using it lots! The goal of today's practicals will be to try and develop a sense of familiarity with git and the most commonly used git workflows. In addition to this, we'll use git and GitHub to get ourselves set up with the codebase for the tutorials going forward.

Git Basics

We'll start by practicing the git basics in a totally clean, sandbox area. Refer back to the slides for info on how to run some of these commands

1. Init a repository called 'hello'
2. Create a text file in that repository called 'hello.txt'
    - Hint: you can use vim to create and edit a text file on the command line… You may have to google how to exit vim once you've used it to make your text file.
3. Add the text 'Hello world' into that text file
4. Check the status of your git tracked files using: `git status`
    - This command will tell you which branch you're on, which files are being tracked, and whether you have added any files to the staging area.
    - For now, you should see your 'hello.txt' as an untracked file
5. Add 'hello.txt' into the git staging area
6. Recheck the status
    - You should now see the file under 'Changes to be committed:'
7. Commit the file
8. Create a new branch named 'mood'
    - So far all changes have been made into the master branch. Let's make some changes in another branch
    - You can check how many branches there are using: `git branch`
9. Add a new line of text in 'hello.txt' to 'I'm feeling good!'
10. Stage and commit this change
11. Merge branch into master
    - You're always merging from another branch into the branch you're currently on. So to do this you'll have to checkout back to master
    - In this case, git was able to automatically merge the two branches
12. Delete branch 'mood'

If you're feeling unsure about any of these, have a play around. Make more branches, make more files, do more merges, use `git status`, `git branch`, or even `git diff branch_name` to see what's going on.

GitHub Basics

1. Sign up to GitHub (if you haven't already)
2. Create a new empty repository on GitHub
    - Refer to: https://docs.github.com/en/get-started/quickstart/create-a-repo for details
3. Link your local copy with the remote repo
    - You need to copy the remote url from GitHub
    - Then run: `git remote add origin <REMOTE_URL>`
4. Push your local copy up to the remote via the command line
5. Make a new branch locally and push
6. Open a pull request on GitHub to merge this new branch into main/master
7. Call us over to review!

Setup the SER Project

1. Clone the repo at https://github.com/tdfirth/ser
2. Use the README on GitHub to find out how to create and activate a conda environment for the project, install the dependencies, and install the codebase as a package
3. Setup cli
    - Under ser/bin you will find a file called cli.py
    - This will be our entrypoint into the rest of our code
    - We will use this, and only this, to perform training and inference
    - We'll build this up over the next few days but for now, note the function `train()` and how it's stitched together with the Typer library.
4. Add another module in ser called 'infer.py'. Add a function in there which prints 'this is where the inference code will go'. Import it and call it from the infer command in cli.py.
5. From the command line, run: `ser infer` and `ser train`

Challenge

1. Create and solve a merge conflict
    - Git can't always automatically merge changes between two branches, in those cases you have a merge conflict: https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts
2. Clone someone else's repo and communicate to each other via branches.