Contact Information:

- Name: Pallav Shah
- Email: pallavshah.ps@gmail.com
- Phone Number: +91 78425 03055
- IRC nick: pallavshah, pallav
- tatoeba.org username: pallavshah
- Time zone: GMT + 5:30
- Natural languages: English, Hindi and Gujarati

Project Details:

Introduction:

As mentioned on the ideas page, the current site has went through a couple of crashes and instabilities and to handle such issues there is a need for multiple administrators. To ease the task of administrators and to make recovery quick and easy, there is a need for some administrative scripts that automates most of the tasks. Since the current repository lacks proper documentation for the initial set up, it becomes very difficult for a new user to set up a development environment. So the project's main aim is to create scripts that simplifies the set-up task and along with that a few supporting scripts that can perform backup, restore, export, import, etc. easily. Also, in order to make recovery from a crash easy and smooth, an ansible based solution is to be created that could help the production server recover from a crash back to its latest stable version.

Along with these administrative scripts, the project extends to implement/improve some export scripts. The existing scripts do not allow on the fly export of dumps and the dumps also lack some useful information. So the aim is to get over with these short-comings as well.

Deliverables:

Specifically, an playbook that can set-up each of the following tasks (roles) will be implemented:

- Backup
- Restore
- Importing data
- Exporting data
- Getting external services up and running
- Updating the production site from the repository
- Restoring to last stable (or n-th stable) state
- Deployment on a real server from scratch
- Deployment on a development machine from scratch
- Adding new languages
- De-duplication of sentences
- Indexing sentences
- Monitoring the server and logging load and activity (subject to time)

The extra features that can be expected from the modified export scripts specifically are:

- Dumps with all the necessary information (which is missing in current scripts)
- Incremental dumps (speed advantage)
- On the fly dumps (database availability advantage)

Each of these scripts will be properly documented. I intend to write all these scripts in python or bash.

Implementation:

This section describes a step-by-step process of how I plan to work on this project through the summer.

I'll be dividing the whole task in five phases, the details of each are as follows:

- Phase 1: This phase deals with installing the CakePHP version of the Tatoeba web app as described here: http://en.wiki.tatoeba.org/articles/show/install-tatoeba-php. This includes installing and setting up tools like Apache, PHP 5.3, MySQL and git. After that creating the database and importing all the necessary data into it will be the next step. Once the data is imported, and the site is up and running, creating a dump of the database and importing it back would be the next move. Once this is finished scripts for the above tasks will be written down. The following scripts will be ready post this phase:

  1. Import scripts
  2. Export scripts
  3. Basic development server deployment script

- Phase 2: This phase involves writing scripts for setting up all the external services used by the Tatoeba website. That includes configuring sphinx and indexing, setting up and configuring suggestd, imagick, furigana and sinoparsd. Following these will be a single script created that will start up all the external services with one command. After that, next step would be to add them to the deployment script. So the deployment script would become quite solid now and would be enough to set up a proper development server.

- Phase 3: In this phase I'd be dealing first with handling git updates (i.e. problems that could occur after a git pull to the configurations and the scripts). Once this is done, I'll be writing down the backup and restore scripts for regular automated backups of the database as well as the configurations and easy restores in case of crashes. This would end the third phase.

- Phase 4: This phase deals with writing scripts for creating virtual development environment and also to have it set-up and configured as a development server automatically. The tools that will be used are vagrant and ansible. Firstly, in order to set up the whole server, I'll write a single ansible configuration that integrates all the automated scripts (involving the set-up and configuration of the server). Ansible uses its playbooks language for configuring servers. So, integrating means porting

all the scripts to the playbook (which shouldnâ€™t be that difficult, given the simplicity of the language).

Next, a vagrant configuration which uses the ansible configuration and sets up the virtual environment with all the necessary specifications. As far as the development server goes, there is no need of using ansible, as the dbserver, webserver, backupserver etc. would all be the same machine (generally, the localhost). So all that could be done by creating a simple vagrant box and using the development script to set-up the tools. For production server however, ansible is very useful as it not only allows easy setup of different machines but it also has module for getting the right code from git and configuring the server using that. So, as a part of recovery scripts, script to recover to a particular state determined by a git commit, would also be provided.

- Phase 5: Once the virtual environment is done, the final phase would be to write down the remaining scripts for adding new languages, de-duplication of sentences etc. And then, next step would be improving the export scripts to include features such as incremental and on the fly dumps and inclusion of any missing but important information in the previous scripts. The documentation of all the written scripts would be the last step. Finally, if time permits, I'll be going through nagios and other monitoring services in order to set up automated server monitoring and logging load and activity.

The different phases mentioned above involve merging of scripts at many places. I plan to do this by writing a single script that can call other scripts with proper parameters. This would simplify the complex task of merging and would also keep script corresponding to each script separate.

Testing Phases:

I'll be doing the testing in 3 phases so as to continue smooth development of scripts without worrying about breaking the already written ones. The three phases are as follows:

- Phase 1: This phase comes after the phase 2 (see previous section) of main development task. This would contain automated tests for import and export scripts and testing of the scripts written to set-up, start and stop the external services.

- Phase 2: This phase, scheduled after phase 2 of main development, will involve testing the backup and restore scripts. Tests involving fake crashes will be done to check the reliability of these. After that, ansible and vagrant configuration scripts will be tested.

- Phase 3: The final phase of testing, which comes in the end will first have the remaining scripts tested through automated testing and then the whole development and production environment will be tested by setting them up individually and performing (and validating) each of the tasks that the scripts are intended for.

Each of the testing phases last a week long at least in order to give time for repairing any broken module that failed during the tests.

Maintenance:

I plan to write these scripts such that they require little or no maintenance. However, as no such code exists, the main maintenance that would be required will be during the update of external services on which the app depends. Also, ansible playbooks might also require some updating now and then as the changes occur on the website.

Technical risks and support:

The single major problem that I could see during the implementation is getting lost in tatoeba's code. Although, the project mainly involves writing administrative scripts that mostly do not interfere with the code itself, but for the import-export tasks and small tasks such as adding new languages, de-duplication, etc. a deeper understanding of the code base is required. And hence without proper support and guidance from my mentor it would be very difficult to get these scripts done. Therefore, this is the area where I'd be requiring most of the guidance from the mentor.

Schedule:

Community Bonding Period:

During this period I intend to talk to my mentor and know as much as I can about tatoeba and its code. Along with that I plan to work out an informal schedule for bi-weekly evaluations of my code so that I get to know I am delivering what is expected and he/she gets to know whether I am getting the work done or not. I also plan to communicate and bond with other people of tatoeba so that I feel more familiar to them during the project period as well as after that.

Timeline:

Here is a very brief and rough week-by-week breakdown of all the tasks as I plan to do them :

- Week 1: Understanding Tatoeba app; setting it up; scripts for installing and configuring must-have tools; getting the first (basic) deployment script ready.
- Week 2: Import and export scripts with improvements; combining and documenting the scripts written till now. (End of phase 1).
- Week 3: Understanding external services and tools used by Tatoeba (going through the documentation); scripts for sphinx index and imagick.
- Week 4: Scripts for configuring suggestd, sinoparsed and furigana; combining scripts for external services (End of phase 2).
- Week 5: Documenting scripts; testing phase 1; catching up with remaining tasks.
- Week 6: Handling git updates; automate backup of the whole database and configuration files; restore scripts. (End of phase 3)
- Week 7: Virtual environment; Understanding ansible and vagrant; setting up ansible; integrating scripts within ansible.
- Week 8: Setting up vagrant; configuring vagrant to use ansible playbooks; automate the whole procedure (End of phase 4).
- Week 9: Documenting the scripts (and configuration files); testing phase 2; catching up with backlogged tasks.
- Week 10: Scripts for remaining tasks (adding new languages, de-duplication, etc.); Improve export scripts; understanding of nagios, cacti, munin; work out a monitoring and logging system for the server; scripts for the tool. (End of phase 5)

- Week 11: Catching up; Documentation of remaining scripts; testing Phase 3 (final).

- Week 12: Finish testing; preparation for submission; informal review from mentor; final touches and submission.

Other details:

Although I have tried to make the schedule as elaborate and well-thought as possible, but it is quite possible that while working on it I may get ahead or behind the schedule. If I get ahead of schedule, I plan to spend the extra time on more rigorous testing as I haven't been able to give much time to it due to the enormity of the tasks that are needed to be done. If I get behind schedule, even after having some catching up periods set up, instead of finishing all the tasks hastily, I'll try to prioritize the tasks remaining and as much as I can without compromising the quality. For that same purpose only, I have kept the task of writing scripts for a monitoring system as subject to time availability. And since I plan to stay as a permanent contributor for tatoeba, I'd be more than happy to finish the remaining task(s) post deadline.